DOCUMENT RESUME

ED 122 856 IR 003 486

AUTHOR Offir, Joseph
TITLE Adaptive Computer-Assisted Tutorials: A Cybernetic Approach Optimization with Finite-State Machines.
INSTITUTION Navy Personnel Research and Development Center, San Diego, Calif.
REPORT NO NPRDC-TR-76-33
PUB DATE Mar 76
NOTE 62p.; Technical report for the period March 1974 - January 1975

EDRS PRICE MF-$0.83 HC-$3.50 Plus Postage.
DESCRIPTORS *Artificial Intelligence; Computer Assisted Instruction; *Computer Oriented Programs; Cybernetics; Educational Research; Input Output Analysis; Instructional Systems; *Instructional Technology; Learning Theories; Mathematical Models; Models; Problem Solving; Training; *Training Techniques
IDENTIFIERS Adaptive Control Systems; *Finite State Systems; Navy

ABSTRACT

This paper presents the concepts of a computer-directed system to improve human performance in structured learning situations. Attention is focused on finite-state systems in order to provide a systematic method for constructing training systems and to assist in analysis of problem solving and curriculum planning. The finite-state approach allows structural relationships among instructional units relative to individual problem solvers to be captured. At this time a fully adaptive system that improves its own performance with use requires further study and development, but it is possible for computer-directed systems that generate and select instructional actions to be used in both training and experimental situations. (CH)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. Nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

ED122856

NPRDC TR 76-33

March 1976

ADAPTIVE COMPUTER-ASSISTED TUTORIALS: A CYBERNETIC APPROACH OPTIMIZATION WITH FINITE-STATE MACHINES

Joseph Offir

This research was supported jointly by the Navy Personnel Research and Development Center and by the Advanced Research Projects Agency of the Department of Defense.

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U. S. Government.

Reviewed by
John D. Ford, Jr.

Approved by
James J. Regan
Technical Director

U S DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

Navy Personnel Research and Development Center
San Diego, California 92152

IR 003 486

FOREWORD

This research and development was conducted in support of Exploratory Development Task Area PF55.522.002 (Methodology for Developing/Evaluating Navy Training Programs). The work is jointly sponsored by the Advanced Research Projects Agency and the Navy Personnel Research and Development Center (NAVPERSRANDCEN). It is the first in a series of reports concerning the development of computer-based adaptive control systems for training. The results are intended to provide the theoretical underpinning needed for the development of a system that can generate instructional sequences, and measure, evaluate and improve a trainee's performance as well as its own performance. The availability of adaptive techniques is expected to enhance computer aids in other areas, such as decision making in command control and in psychological experimentation

The author expresses appreciation to Drs. John Ford, Jr. and William Montague, NAVPERSRANDCEN for their continuing support and encouragement, and to Professor William Batchelder of the University of California, Irvine, for his insightful comments. Appreciation is also extended to Mr. John Wolfe and Dr. J. D. Fletcher of NAVPERSRANDCEN and to Dr. Richard Wollmer of the University of Southern California.

J. J. CLARKIN
Commanding Officer

SUMMARY

Problem and Purpose

Computer-assisted instructional (CAI) systems have been in use for over a decade. This period has been marked by great optimism and an ever-growing demand for electronic gadgetry. Nonetheless, the development of CAI has not progressed beyond the stage of infancy; the optimism about its capabilities seems disproportionate relative to the small number of real conceptual breakthroughs in the field.

One source of optimism has been the flurry of research activity in linguistics and artificial intelligence (AI), which many people have hoped would inspire CAI systems possessing flexible dialogue capabilities in natural language. However, there are less than a handful of operational CAI or AI systems that have any conversational capabilities. The task of programming such a system is immense, and the resulting program requires a sizable chunk of memory. More important the few programs that do exist incorporate only limited behavioral goals for the user, structural descriptions of the subject matter, models of the user's behavior with respect to the subject-matter, and performance criteria. Hence, they lack explicit and direct ways to measure, evaluate, and improve the user's performance.

Only a few theoretical studies have attempted to deal with the problem of improvement or optimization in the context of CAI systems, and none gives a comprehensive treatment of the subject. The present paper attempts to fill this gap.

Approach

The approach taken here is practical, purposive, and prescriptive: it concerns the development of computer-directed systems to improve human performance in structured learning situations. The intent is to construct a system capable of decision making; such a system must generate and select optimal instructional actions, under resource constraints, that will lead the student to perform in some desired manner. For example, the system must be able to decide whether to generate a particular display, test knowledge of a particular concept, terminate an interactive session, etc. Since the choice of the best action depends on the quality of the information available, the system must also be able to determine the sequence of experiments needed in order to gather the best information.

A system designed to learn while it earns is called an adaptive system. The behavior of an adaptive system is determined by our objectives and constraints and an evolving model of the user. Initially, the user is represented in the system as a "black box" or as a family of finite-state models. It is assumed that the user's performance is a probabilistic function of his action states. The system, however, does not know the exact internal structure of the user-model and cannot directly observe state occupancy or state transitions. Consequently, the system chooses a course of action that allows it to construct or select a parsimonious model of the user; any version of the model must account proabilistically for all

possible stimulus/response sequences. In addition, the system may determine retrospectively, under various conditions, the initial state of the user, or may predict his final state after observing only a minimal number of stimulus/response pairs.

The following rudimentary concepts are used as building blocks: states, deterministic and stochastic systems, finite-memory systems, input-output analysis, state and system equivalency, reduced forms, state-calculable systems, experiments, and optimal and adaptive control.

This paper attempts to provide as rigorous a presentation as possible without undue emphasis on mathematics. Consequently, attention is focused on finite-state systems only, and applications where the sets of stimuli and responses can be represented by a finite number of symbols. It is hoped that sufficient insight is provided to develop a broader conceptualization that will apply to a wider range of applications in CAI, psychological experimentation, and man-computer systems.

Conclusions and Recommendations

Computer-directed systems that can optimally generate and sequentially select instructional actions are needed in training and experimental situations. The usefulness of such systems is predicated on their ability to formally capture the structural relationships among conceptual units vis a vis each individual problem solver, decision-maker, or a trainee, as the case may be. The finite-state approach provides a systematic way for contructing training or aiding systems as well as for analyzing curriculum planning, evaluation and problem solving in general. Further work should be undertaken to develop several adaptive systems in training or decision-making environments in order to determine their practicability and limitations.

CONTENTS

	Page
FOREWORD	v
SUMMARY	vii
INTRODUCTION	1
Purpose	1
Problem	1
Background	3
METHODS OF TRAINING STUDENTS TO SOLVE FORMAL PROBLEMS	7
Formal Problems and Their Solutions	7
Tutorial Processes in Problem Solving	13
A Taxonomy and Brief Survey of CAI Systems	15
Probabilistic Systems	18
Models	18
Optimization, Adaptation, and Individual Differences	19
A Toy Example of Generative Remedial Feedback	21
A General Optimal-Control Problem	22
FINITE-STATE SYSTEMS: MODELS, EXPERIMENTS, AND CONTROLS	25
General System Theory	25
Inputs, Outputs, Systems and Their States	25
Identifiability of Deterministic Systems	29
Deterministic Finite-State Systems: Equivalence and Distinguishability	30
Experiments with Finite-State Deterministic Systems	33
The Structure of Input, Output, and Cost	39
Optimal Control Problem of Finite-State Systems	40
STOCHASTIC FINITE-STATE SYSTEMS: MODELS AND OPTIMAL CONTROL	42
Definitions and Basic Relations	42
Special Classes of Stochastic Systems	45
Identifiability of SFSS With Finite Memory	50
Optimal Control of Stochastic Finite-State Systems	52
SUMMARY AND CONCLUSIONS	53
REFERENCES	55
DISTRIBUTION LIST	59

CONTENTS

Page

LIST OF FIGURES

1. Deterministic solution paths digraph 11
2. Adjacency matrices of solution paths 12
3. A tutorial system for problem solving. 14
4. A schematic diagram of a general control system 22
5. An object with a finite memory 25
6. A system of two objects in series 26
7. The state diagram corresponding to Table 2 28
8. Two weakly equivalent systems 30
9. State-diagram 32
10. A system M and its reduced form M' 33
11. Adaptive diagnostic experiment 36
12. Multiple diagnosing experiment vs. a simple experiment 37

LIST OF TABLES

1. Input-Output Pairs of Length 3 26
2. Bundles of I/O Pairs Listed in Table 1 28

INTRODUCTION

Purpose

This report outlines a formal approach to computer-based adaptive tutorial systems. In particular, it analyzes the components a tutorial system must have in order to improve dynamically the performance of an individual trainee, with respect to prechosen criteria, in a sequential process of formal problem solving.

Such a system must analyze the trainee's responses and evaluate his performance, determine difficult and easy solution paths, generate feedback to the student dynamically, and stop tutoring when desirable. The system should generate feedback without requiring laborious prespecification of the text or enumeration of all anticipated responses, and it should improve its own performance over time.

Problem

The United States Navy faces a serious challenge to its long-established superiority because of the proliferation and increasing sophistication of the world's military technologies. The ability to meet this challenge by advancing and manning its own technologies is hampered by the economic realities of our time. Since readiness depends on the effectiveness and efficiency of our training processes, we must continuously prepare a large number of proficient personnel in a number of disciplines and at a minimal cost. Consequently, our training procedures must be made amenable to economic scrutiny by employing operational methods such as cost-efficient quality control.

In general, we need to consider two types of interrelated issues, administrative and instructional. Administratively, we have to translate our sociotechnological needs into educational objectives and, in turn, determine optimally who should be trained, in what, when, and why. Instructionally, we have to develop systematic methods to improve the performance of an individual trainee in a specific administrative context.

Many administrative problems can be solved by invoking operations research techniques, such as queuing and scheduling methods, for resource allocation. By using these techniques we can determine, for example, the time periods when various training resources should be made available to different trainees, and the order in which the resources should be allocated. Since training resources are scarce and the need for competent personnel is usually urgent, optimal administrative procedures should be designed to maximize the number and the quality of the trainees, while simultaneously minimizing the time expended by the trainers and the trainees.

It is mathematically unfortunate that human beings, unlike objects molded en masse in a production line, possess nonstationary individual differences that must be taken into account in such an optimization process. One consequence is that, even if our primary goal is to maximize the collective performance gain of a trainee group, we may have to adapt our training to each and every individual in the group over a period of time. It is important to recognize, therefore, the close reciprocity between macro-administrative decisions and micro-individual variables.

Furthermore, we cannot make administrative decisions without evaluating the quality of instruction. That is, we must be able to measure gains in behavioral performance relative to the expenditure of training resources. It is easier to define and measure the latter than the former. Training costs are associated in a straightforward way with the preparation of display material, data analysis/evaluation, and expenditure of man-hours by the trainers or the trainees. In contrast, although behavioral performance can be expressed as a function of correct responses or response time, it is very difficult to formulate more sophisticated indicies of performance gains based on the cognitive processes of problem solving and improvements in these processes. Measurements, and hence training problems, are complicated because trainees may possess different initial knowledge, may learn at different rates, and may interact with the material in different ways.

Unfortunately, curriculum designers and evaluators often neglect these issues. They may compare pre-and post-test scores averaged over different trainees and training material in order to apply statistical tests of significance and retroactively evaluate hypotheses about group differences, because it is static rather than adaptive. Such methods cannot provide a basis for modifying and optimizing instruction until instruction has terminated. By that time, of course, resources have already been allocated and expended.

Such problems have not been overcome by increasing use of computers for instruction. Originally, computer-aided instruction (CAI) was expected to provide a less costly alternative to many forms of individualized instruction. Most of these expectations have not been realized because we have adhered to static methods of planning and evaluation. Consequently, CAI programs have come under fire for being not only plodding, inflexible, and unimaginative, but also expensive; each program requires considerable and separate effort. Computer-based instruction is where farming was 200 years ago, when a dozen farmers were needed to feed one nonfarmer. It is quite inefficient.

On a cursory inspection, the criticism appears to be justified. It is important to realize, though, that the problems of CAI are very complex and stem from the absence of a good theory of cognitive organization.

To quote George Miller (1974, p. 97): "We have not begun to realize the intellectual potential that computers could provide, and... will not make very rapid progress toward that realization until we have formulated better psychological theories of cognitive organization." However, it seems unlikely that we will devise a satisfactorily comprehensive and coherent theory of learning and teaching in the near future.

To summarize, administrative decisions cannot be made independently of instructional decisions, and vice versa. To make optimal instructional decisions, one needs dynamic methods of evaluation and decision making. In the best of all possible worlds, these methods would be based on a comprehensive theory of cognitive organization. Such a theory does not now exist, and its absence continues to handicap the development of CAI systems. Nevertheless, there are decision-theoretic techniques that can be used to improve computer-assisted training programs. These techniques apply when trainees are taught to solve a specifiable class of problems and there is at least one efficient algorithm for reaching a problem solution in each instance. These techniques are discussed in succeeding sections of this paper.

Background

Before proceeding with a detailed discussion, it is appropriate to outline the background and philosophical underpinning of the approach taken. In order to develop successful adaptive tutorial systems we must impose a metric on training sequences as well as measure performance and costs. This should be done by deriving the interrelationships among existing psychological theories of learning and cognition. The first order of business is to reconcile differences between mathematical theories of learning (e.g., Atkinson, Bower, and Crothers, 1965) and information-processing theories of cognition (e.g., Newell and Simon, 1972). This reconciliation can be achieved under the rubric of mathematical system and cybernetics theories (e.g., Kalman, Falb, and Arbib, 1969; Arbib, 1972).

Certain aspects of mathematical learning theories are very attractive because they allow quantification of psychological entities, such as learning rates, by means of a small set of parameters. However, mathematical learning theory arose out of the behaviorist tradition in psychology and fails to explain crucial aspects of cognitive processes or describe the structure of knowledge.

Chomsky (1959) dealt a blow to the behaviorist school and suggested a formalism for explaining linguistic structures in terms of generative grammars. His theories, however, explicate the formal properties of linguistic competence but do not describe the actual performance of natural language users. Subsequent attempts to turn abstract linguistic models into useful empirical descriptions of behavior have failed.

Nonetheless, Chomsky's theories of competence have produced important changes in our thinking about complicated behavior such as problem solving. Chomsky and others have shown how formal rules, called grammars, can be used to characterize classes of symbolic strings, called formal languages. Certain computer languages can be characterized in the same way. In many instances, a mathematical device, called an automaton, can be used to recognize or generate the strings that belong to a given language. Whereas most mathematical learning theories ignore structures intervening between stimuli and responses, automata models can explain how different outputs may be generated from the interaction of different inputs with the internal states of a device. Instead of merely examining whether learning has or has not taken place on a particular trial, these models explicate the structural contingencies in the output of the device as a function of the contingencies in the input and the structure of the device (Offir, 1973).

There is a connection between these formal notions and information processing models of complex behavior, such as the General Problem Solver, (GPS) (Newell, Shaw, and Simon, 1959) and the Test-Operate-Test-Exit hierarchy model (TOTE) (Miller & Chomsky, 1963). TOTE hierarchies and GPS operate in essentially the same way. Briefly, a TOTE unit consists of two parts: (1) a test to see if some situation matches an internally generated criterion, and (2) an operation that is intended to reduce any differences between the external situation and some internal criterion (Miller & Chomsky, 1963). TOTE hierarchy, like a computer program, is a process that can control the order in which a sequence of operations is to be performed. Any TOTE hierarchy corresponds to a finite automaton (i.e., an automaton with only a finite number of internal states) (Miller & Chomsky, 1963; Suppes, 1969). Furthermore, there is an intriguing similarity between the GPS/TOTE concepts and the cybernetic scheme of transforming an initial pattern to some desired form (Pitts & McCulloch, 1947), e.g., transforming $3y + 5 = 20$ to $y = 5$.

Consequently, the focus here is on finite-state models in a search for representations that will account for both the structure of a task and behavioral processes for manipulating this structure by means of formal operations. Since novice problem solvers usually exhibit unpredictable behavior rather than deterministic behavior (which is more characteristic of experts or teachers), this paper concentrates on probabilistic versions of finite-state automaton to account for the uncertainty. (Automata and machines are considered synonymous in this section; in following sections, they are precisely defined.)

The engineering literature contains many applications of probabilistic, finite-state machines to communication, reliability, and pattern recognition problems (see Paz, 1971 for references), but psychological/educational literature neglects such models. (An application to arithmetic problem solving is described by Offir, 1973.) It is interesting to note that although the use of Markov models is growing in the behavioral/social sciences, most investigators are unaware that Markov models constitute only a special case of probabilistic, finite-state machines.

After discussing methods of training students to solve formal problems, this report describes the formal properties of finite-state models and indicates how they may be used to represent a class of problems and associated problem-solving behavior. Finally, it combines the descriptive aspects of these models with dynamic-programming methods and outlines how to construct an adaptive tutorial system that has a desired set of characteristics.

METHODS OF TRAINING STUDENTS TO SOLVE FORMAL PROBLEMS

Formal Problems and Their Solutions

Most approaches to problem solving are based on the expert's knowledge about the problem's structure. Less attention is given to the prerequisites for learning rules to solve problems expertly, and almost no formal evidence exists about the process of acquisition, i.e., the kinds of training that a beginning problem solver must undergo in order to gain expertise. The "New Math" is a good example: it emphasizes the structure of the subject matter but shows little concern with the process of cognitive acquisition.

Suppose we observe a machine or an organism solving a problem. Formally speaking, we can only record the overt parts of this process that are exhibited externally. They consist of an initial representation (u_1), a terminal representation (v_f), and a sequence of formal operations that transform the problem (u_1) to its solution (v_f). In general, all strings of overt symbols are referred to as input (u_j) and output (v_j). For example, u_1: $3z + 5 = 20$, v_1: $3z = 15$, u_2: $3z = 15$, $v_2 = v_f$: $z = 5$. Often, when the final output is known *a priori*, as in the Tower of Hanoi problem (Nilsson, 1971), it is referred to as the goal, and the intermediate outputs, as subgoals. In either case, a goal is reached or a solution is acceptable if v_f satisfies a set of external criteria. A solution need not be incorrect to be unacceptable; e.g., v_1: $2z = 10$ is a correct output for the problem above but may still be unacceptable because it is not in the required final form. An unacceptable output string, whether correct or incorrect, is called a nonterminal string and is denoted by v_N.

In addition to output strings, almost all problems can assume several unobservable initial representations that may consequently influence the effort and efficiency involved in solving the problem. There are many examples of seemingly difficult problems that become trivial under an appropriate choice of initial internal representations (see Nilsson, 1971, for references). Unfortunately, very little is known about the perceptual-cognitive processes that are required to "look at the problem at the right way," and arrive at the best initial representation. It is conjectured that these processes change as a function of the problem solver's experience; that is, they improve with his ability to recognize and combine separate features into a general pattern and to concatenate individual operations into macro operations, e.g., additions into multiplications.

In building a model of the problem solver, the number of features or variables that can be used to describe his internal encoding of the problem is unlimited. A crucial step in any theory construction is to select only the most salient features. For example, in theorizing about problem solving in arithmetic, there is little reason to expect that the color or the thickness of the printed page will influence the problem solver's performance noticeably. Choosing variables that really matter is an art, therefore, and is not dictated by an exact algorithm.

We still have to define operations. Formal operations should be distinguished from mental operations. Roughly, formal operations are transformations of inputs, which consist primarily of non-terminal outputs (v_i), into subsequent outputs. For example, in "3 plus 4," "plus" represents a formal operation. Mental operations are unobservable steps carried out by the problem solver to accomplish formal operations. Two types of formal operations can be distinguished - primitive and complex. A primitive operation involves an atomic, one-pass, mental operation that can transform a symbol or a string of symbols. A complex operation is composed of two or more primitive operations.

Two factors help determine whether a formal operation is primitive or complex: the problem solver's skill and the length of the operands. For example, most adults know what 3 times 4 is without performing any intermediate steps. Fewer know what 34 times 43 is, and only a handful know what 3456 times 789 is without writing anything down. Still, for many first-graders, the solution of 3 times 4 requires a number of intermediate operations.

A need exists for effective and efficient solution methods for a given class of problems. At first glance, efficiency appears to require as few operations as possible (both mental and formal). One obvious way to minimize the number of intermediate solution steps is to memorize the answer; most problems would require only one primitive operation once we memorized their solution. However, memorizing facts, like storing data in a digital computer, can be more expensive than using many solution steps. The formal analysis of algorithms (Knuth, 1968) shows that efficiency must be expressed in terms of both operations and storage and the tradeoff between them. Ideally, efficient problem-solving training methods should make use of the interplay between memorization and mental operations, and emphasize solution paths of optimal mix.

Unfortunately, too little is known psychologically about such optimal mixes to be of any practical value in training people to solve problems. The question is, then, what can we say about the formal aspects of problem solving? Mathematicians and computer scientists have been concentrating primarily on the following subquestions:

1. Given specifications of a class of initial inputs (U_1), a class of terminal outputs (V_f), and a sequence of operations $d_1d_2...d_L$ from a set D of allowable operations, where d_L is the last operation applied, what output v_f in V_f is such that $d_1d_2...d_L(u_1) = v_f$ for each input u_1 in U_1?

2. Given the specifications of U_1 and V_f, what is a sequence of operations that would transform any input in U_1 to some class of outputs in V_f?

3. Given two problem-solving systems A and B, i.e., (U^A, D^A, V^A, u_1^A) and (U^B, D^B, V^B, u_1^B), where U and V are the sets of all input and output strings, respectively, what are the formal relations between A and B that represent the notions of sameness and similarity, etc?

The answer to the first question can be a computer program. In some sense, we are looking here for the human analogue to computer programs and computational processes. For example, what is (5+3) x 5 ÷ 4?

The second question deals with "rule discovery." In the literature of computer science it is known as the problem of automatic program synthesis (Manna and Waldinger, 1971). Research in this area is still in its infancy. A simple example of program synthesis is the following: Given $U_1=\{5\}$ and $V_f=\{4,14,24,...,94\}$, find a sequence of operations in U_1 that derives the numbers in V_f. By recharacterizing V_f in meta symbols $V_f=\{v_n | v_n=4+n(10)$ such that $n=0,1,2,...,9\}$, it is readily seen that $v_{n+1}=v_n+(5+5)$ and $v_o=(5+5+5+5)/5$.

The third question emphasizes the structure of operations rather than the discovery of operations or the computational process. It may be useful to view problem solving systems in the framework of formal, Chomsky-type languages (Fu and Booth, 1975). In this linguistic analogy, u_1 corresponds to a start symbol, the operations in D correspond to the appropriate production rules of the corresponding grammar, and the output sequences constitute the language.

A full explication of the formal aspects of problem solving requires better understanding of the three subareas we have mentioned. For the present, the discussion is limited to computational aspects of problem solving. Although this approach greatly limits the kinds of problem-solving tasks that can be studied, it provides a means to quantify some problem-solving processes.

Example. Given a class of initial representations of the form $ay + b = c$, where a, b, and c are real numbers, y = ?; for example, $3y + 5 = 10$. To solve this problem, one must apply a sequence of formal operations that will transform, or eliminate, some of the original symbols in u_1 and terminate with a string v_f. In this example, the set of allowable operations D includes the four operations: add, subtract, multiply, and divide. When an operation such as add is applied to an input, we must specify a second operand, i.e., we have to divide the input by something or add something to it, rather than simply divide or add. An efficient solution sequence may look like this:

d_1: subtract b	d_3: divide by a
v_1: $ay + 0 = e$ where $e = c - b$;	v_3: $1y = f$ where $f = e/a$
d_2: add a zero	d_4: multiply by one
v_2: $ay = e$	v_f: $y = f$

Note: We assumed here that $u_{i+1} = v_i$ for all i.

Also e and f result from unobservable mental operations.

Note that of the six symbols appearing in u_1 only "y" and "=" are going to remain unchanged. We could therefore represent the above solution path as follows:

$(a, +, b, c) \xrightarrow{-b} (a, +, 0, e) \xrightarrow{+0} (a, \phi, \phi, e) \xrightarrow{1/a} (1, \phi, \phi, f) \xrightarrow{x1} (\phi, \phi, \phi, f)$, where ϕ corresponds to a blank entry. The first string represents the initial input and the last one the terminal output. The labeled arrows denote the corresponding operations. Another solution path which is not as efficient as the first one because it requires additional divisions can be represented as follows:

$(a, +, b, c) \xrightarrow{1/a} (a', +, b', c') \xrightarrow{-b'} (1, +, 0, c'') \xrightarrow{+0} (1, \phi, \phi, c'') \xrightarrow{x1} (\phi, \phi, \phi, c'')$, where $a' = a/a$, $b' = b/a$, $c' = c/a$ and $c'' = c' - b'$. In general, any input or output string may be represented by the four-tuples (w_1, w_2, w_3, w_4). If we allow then each symbol w_i, $i \neq 2$, to assume two values--a real number or a blank--and w_2 to be "+", "-" or "ϕ", we may have at most 2x3x2x2 = 24 possible strings. The solution path is determined by the order of operations chosen.

Pictorially, the solution paths can be represented by a labeled directed graph (digraph), such as the one in Figure 1. The two paths start with the same input and terminate with the same multiplication operation, d_L = multiply by 1. Generally, our labeled digraph may have several initial and terminal nodes, and its solution paths may cross and share common solution segments.

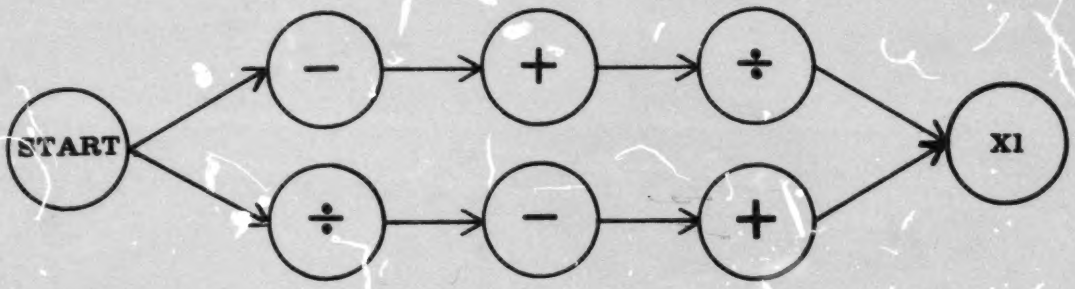


Figure 1. Deterministic solution paths digraph.

A representation that is more useful algebraically can be given by the adjacency matrix that corresponds to the labeled digraph (Figure 2). The adjacency matrix is a square matrix with L rows and L columns, where L designates the number of operations. Its entries, a_{ij}=1, indicate whether the operation d_j immediately follows operation d_i. No operation follows d_L, the last operation.

		d_L	d_3	d_2	d_1
	d_L =	1	0	0	0
	+ d_3	1	0	0	0
	+ d_2	0	1	0	0
Start →	- d_1	0	0	1	0

(a) Adjacency matrix for the first solution path

		d_L	d_3	d_2	d_1
	d_L	1	0	0	0
Start →	+ d_3	0	0	0	1
	+ d_2	1	0	0	0
	- d_1	0	0	1	0

(b) Adjacency matrix for the second solution path

Figure 2. Adjacency matrices of solution paths

The previous example was selected for its simplicity. More illustrative problems can be drawn from trigonometry and calculus. For instance, there are many ways to evaluate the integral $\int \sin^2(x) \cos(x)\, dx$; however, the direct substitution $\sin(x) = w$, $\cos(x)\, dx = dw$, and $\int \sin^2(x) \cos(x)\, dx = \int w^2\, dw$ is by far the most efficient way to obtain the solution.

To this point, only deterministic and linear solution paths have been considered--deterministic because each operation follows another with certainty, linear because repeated use of the same operation is ignored. To allow for uncertainty and nonlinearity, the zero-one entries in the adjacency matrix can be replaced by probabilities.

For each distinct input, u, there is a corresponding matrix with entries $p_{ij}(u)$, denoting the relative frequency with which d_j follows d_i; the sum $p_{ij}(u)$ over j, for a given i, is equal to 1. Which operation is actually executed is not directly observable. To overcome this obstacle, the $p_{ij}(u)$ entries are partitioned to $p_{ij}(v|u)$, i.e.,

the conditional probability that v is emitted in response to the input u. Whenever d_j follows d_i the sum of $p_{ij}(v|u)$ over v is equal to $p_{ij}(u)$. The probabilistic matrices, $[p_{ij}(v|u)]$, are in fact the machines discussed in this paper--they are called stochastic finite-state machines. Deterministic finite-state automata are a special case of this stochastic generalization. The equivalence of such automata to TOTE hierarchies and, by implication, to the GPS approach, was already mentioned. The reader may also recognize the relation of finite automata to regular, type-3, languages. Interestingly, stochastic finite-state machines can account for many examples of context free, or type-2 languages (for references see Paz, 1971).

Finally, it is important to recognize that "the state of the machine" is associated with an operation rather than with a stage in the problem solution or with the ambiguous notion of "state of knowledge." This conceptualization of a state departs radically from previous interpretations (e.g., in artificial intelligence, see Nilsson, op. cit.; in mathematical psychology, see Atkinson et al., op. cit.). Interpreting states as operations and not as the problem's subgoals, as in the AI interpretation, yields a small and manageable set of states. The interpretation of a state as a learning stage, e.g., the "learned state" or "forgetting state" in mathematical learning theory, may be too metaphysical to be operationally useful.

Tutorial Processes in Problem Solving

The description of a tutorial process begins with a computer metaphor. If a process is well understood, then its salient characteristics can be imitated by a machine. Conversely, using a machine metaphor provides insight into a student-tutor interaction relative to a given subject matter. This type of approach falls properly within the framework of cybernetics, which was defined by Norbert Wiener (1948) as "the study of control and communication in man and machine."

In Figure 3, boxes A and B represent two communicating computers, a tutor and a student, respectively. Their dialogue is marked by interrelated episodes of formal problem solving. At the beginning of each episode, A generates an input signal, u_1, that B is required to solve. Upon reading u_1 and interpreting it into its own machine language, B proceeds to derive a solution by sequentially executing a series of subroutines or operations. The output resulting from each operation is disclosed to A at intermediate stages.

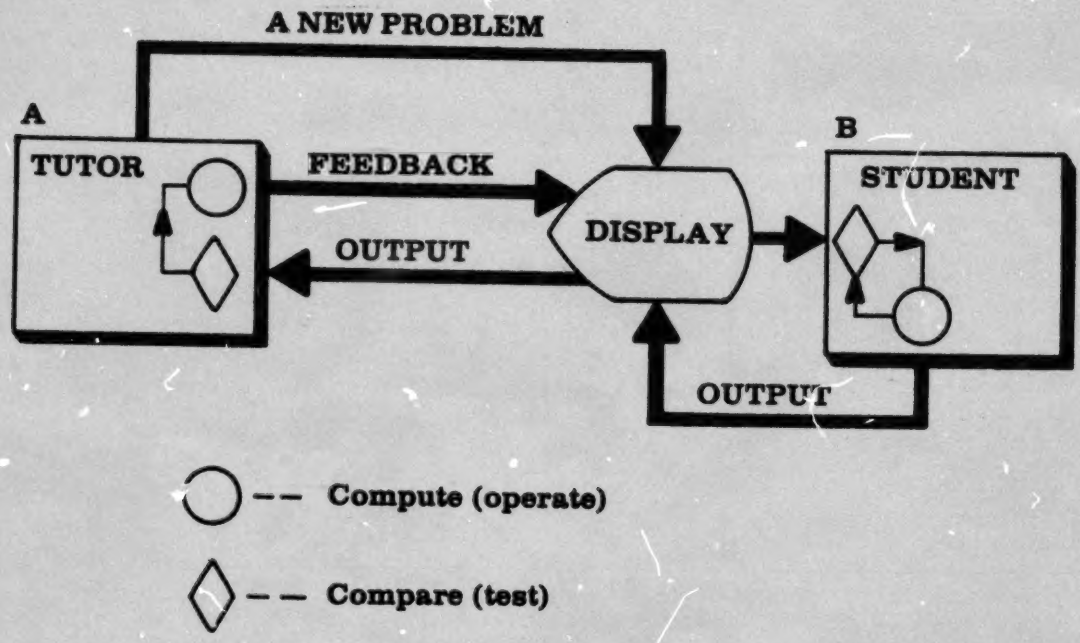


Figure 3. A tutorial system for problem solving

Although B's choice of operations is probabilistic, it is influenced by B's previous output, if any, and additional feedback inputs communicated by A. A feedback signal is generated as a function of the discrepancy between B's output and the output desired by A. B is an unreliable machine, and its output is only a probabilistic function of the operation chosen and the preceeding input and output (see paragraph on Output Independent Machines on page 48).

Unlike the student, the tutor A can compute the shortest sequence of operations, or solution path, that yields the desired output at any stage. It can utilize this knowledge to direct the student. However, A cannot observe directly the exact sequence of internal operations executed by the student; it has to infer it indirectly from B's external behavior. This inference, from external behavior to internal, is complicated by the possibility that two or more internal operations may produce the same output. This is the problem of identification (see page 29).

The saving feature of the situation is that A can adapt to and learn more and more about B's internal selection of operations. A great deal of this learning is done through a statistical process of updating probabilities. By tallying B's output under various conditions, A can estimate the probability that B will select a particular operation and emit a specific output as a result of that operation.

The tutor, A, generates problems and related feedback designed to optimize the student's output. By continuously monitoring this output, the tutor program can compute the effect that various combinations of problems and feedback have on the student's choice of operations and output. The ultimate purpose is, of course, to modify the probabilistic behavior of the student and cause him to choose operations and responses deterministically and in some optimal fashion. A can appropriately be called the Great Rule Inducer (Simon & Lea, 1974).

A working premise is that problem-solving processes depend on cognitive context and the structural contingencies among stimuli. This study does not speculate about learning in the associationistic sense of pure probabilistic reinforcement schemes that are independent of stimulus-valence (see for example, a discussion on linear reinforcement schemes by Atkinson et al., op. cit.). Neither does it consider the student as a learning machine, an approach adopted by control engineers (for a survey, see Narendra and Thathachar, 1974). In representing a student, we must try to capture the tangible performance characteristics associated with the execution of an algorithm rather than describe abstract learning using contextless concepts such as punishment or reward.

A Taxonomy and Brief Survey of CAI Systems

This section reviews terminology that is generally used to classify CAI systems. The taxonomy is not exhaustive and the classes of systems are not disjoint, i.e., some systems may belong to more than one class. The objective is to gain perspective and appreciation of various features of CAI systems.

First, a distinction is made between model-driven and model-free systems. A model-driven system contains a formal representation of one or more of the following:

1. A set of training material, denoted by M_m.

2. A set of teaching operations representing the tutor, denoted by M_t.

3. The student's problem-solving behavior, denoted by M_s.

In model-free systems, there is no explicit formalization of the relations among the components of the system. Model-free systems consist of structurally independent frames that need to be preprogrammed individually, rather than be generated on line. Such systems have been called frame-oriented (Carbonell, 1970). To the extent it exists, the branching logic in frame-oriented systems is predetermined just as it is in a preprogrammed text. The feedback in frame-oriented systems is anticipatory. In addition, subsequent displays rarely depend on the student's previous responses. A system with these characteristics is also labeled response

insensitive. The best known examples of frame-oriented systems were developed at Stanford University in the 1960's and include drill and practice programs in arithmetic, logic, reading, etc. (Suppes and Morningstar, 1972).

Most early efforts to formalize aspects of CAI systems were motivated by a desire for greater precision in making instructional prescription and were centered on a mathematical learning model of the student, M_s. The Chomskian influence, however, caused most researchers to shift their attention from models of performance based on abstract learning curves to models of language.

Since then, information processing and cognitive theories of artificial intelligence and psychology have been gaining momentum. Their impact on CAI has been considerable, not so much in advancing the state of real operational systems, but in influencing the thinking and expectations of researchers in the area. As a consequence, the focus has shifted once more from competence and formal grammars to the dynamics of interactive systems; that is, systems designed to interact with students in a flexible form of English.

New CAI systems center around well organized, cohesive, relational models that delimit the universe of discourse, M_m. In earlier systems, M_m was represented as a semantic network (Quillian, 1969), e.g., a city is a subconcept of a country, a country is a subconcept of a continent, ergo, a city is a subconcept of a continent. This type of representation allows some limited deduction but does not provide for dynamic inference, i.e., the ability to answer questions concerning casual chains of events, transitions, and the reasons for them. SCHOLAR (Carbonell, op. cit.) is an example.

Current thinking, notably advanced by the Artificial Intelligence Group at MIT (Hewitt, 1972; Winograd, 1972), holds that the structure of M_m should be analyzed procedurally; this approach is known as the "procedural embedding of knowledge." Procedural systems are goal directed: roughly, the system generates successive subgoals by evaluating its "distance" at each stage from the desired end-result. Procedures are routines that formalize the process of problem solving in terms of subgoals and actions. This, of course, is reminiscent of TOTE hierachies and GPS.

In some model-driven systems, the flow of conversation is controlled by either the student or the system. A system of this type has been called a mixed-initiative system (Carbonell, op. cit.). Either student or the computer may control the dialogue by selecting a conversational topic and asking questions about it.

Another class of model-driven systems is labeled generative. Its development is motivated by the need to reduce storage and preprogramming requirements. A generative system can be either mixed-initiative or computer controlled, response sensitive or insensitive but it is always model driven. The output of a generative system is not preprogrammed but is derived from a parsimonious set of data and numeric and nonnumeric derivation rules.

When the generators are numeric, display items are generated by assigning numerical values to a few parameters, e.g., assigning values to a, b, and c in $ay + b = c$ or $az^2 + c = 0$. These parameters can be assigned random numbers, but, preferably, should be determined as a function of the student's performance history.

A nonnumeric generator consists of a single inference rule that specifies a relationship among objects (O), properties (P), and values (V) in a network. In order to generate a problem, one can associate three different types of generators with each triplet of the form (O, P, V): (O, P, ?), (O, ?, P), (?, P, V). For example, (California, largest city, ?), (California, ?, Los Angeles), and (?, largest city, Los Angeles). Examples of generative systems are described by Wexler (1970). A toy example of generative feedback is described on page 21. The reader should note that mixed-initiative systems must be generative in some sense.

When a class of systems is both generative and response sensitive, some researchers (e.g., Goldberg, 1973) make a further distinction between anticipatory and adaptive response analysis. In anticipatory systems all possible response categories are prespecified (Wexler, 1970). In adaptive tutorials, the system must be able to analyze any response dynamically. In adaptive systems, the remedial material is generated by the same routine that analyzes the errors. Our toy example (page 21) illustrates this with a simple arithmetic tutorial. The definition of adaptiveness is more restrictive; in addition to the above requirements, the system must improve its tutorials overtime by teaching more students.

Although model-driven systems can provide interactive capabilities, the cost of developing such systems is very high because of the enormous programming effort, storage requirements, and the program complexity involved. The paucity of operational model-driven systems tends to reflect this point. Lighthill (1972), in his state-of-the-art report on artificial intelligence, used the term "combinatorial explosion" to describe the major cause for these difficulties. This term refers to the fact that a small increase in the size of the subject matter tends to increase exponentially the corresponding computational effort.

This paper is primarily concerned with the optimization aspects in CAI. An underlying premise is that, in order to improve and measure the efficiency of a CAI system, we need to represent objectives, costs and constraints, and system/student performance in some formal, quantitative manner. Since the magnitude of most of the variables in question is uncertain, it is reasonable to represent them in terms of probabilities and associated statistics.

Probabilistic Systems

Models

The formal uses of probability and its associated concepts are found in the construction of models of random phenomena and in the design of inference and decision-making systems. With the advent of the computer, probabilistic modeling and optimization have played an increasing role in the design of information processing systems. Currently, only limited application of advanced probability methods is found in operational CAI systems despite the fact that the potential for applying optimization methods to dynamic probabilistic systems, such as computer-directed teaching systems, has been recognized for over a decade (Smallwood, 1962).

Several general factors may explain why probabilistic approaches have not contributed prescriptive solutions to behavioral problems. It is increasingly clear that we need to improve our understanding of probability before we can expect to measure it and use it to select goals and performance criteria. Fine (1973) provides a critical insight into the issues involved.

One problem that is often neglected is how to measure probability. We need a process for converting data into probability statements. Most data are not only explicit (e.g., frequencies, sequences) but also implicit (e.g., beliefs or opinions, the way we select the explicit information). Thus, a difficulty arises because we usually cannot specify algorithms for converting implicit data into probabilities. A growing number of psychological studies are devoted to this issue (e.g., Tversky and Kahneman, 1973).

The definition of probability as a finite relative-frequency answers the measurement question, at least for random phenomena that are unlinkedly repeatable, i.e., observations that are stochastically independent. Unfortunately, this definition of probability provides only a narrow scope for decision making and inference. Finite relative-frequency is descriptive of past behavior in an experiment but is difficult to justify as being predictive of future behavior; the frequency in the first N trials is usually not the frequency in the next N trials. Furthermore, not all random phenomena are amenable to analysis in terms of arbitrary repetitions. If we look closely, we see that we never exactly repeat any experiment.

Behavioral scientists and psychometricians have been using almost exclusively the relative-frequency interpretation of probability and associated sampling methods for inference. In order to apply these methods correctly, the data-generating process has to meet certain assumptions. For example, in order to estimate parameters, it is usually necessary to assume that the parameters remain unchanged over time and that the observations are identically and independently distributed. The interpretation of the last two assumptions in behavioral contexts ignores the issues of individual or stimulus differences and of contextual contingencies, respectively. Such assumptions are rarely met in dynamic situations like learning. It has been shown that even in simple laboratory paradigms there are important departures from the assumptions that influence the validity of the conclusions (Offir, 1972).

Thus, the use of statistical sampling methods restricts us to a post-hoc, one-shot, summative description of the data. In behavioral settings, Bayesian analysis provides a more realistically interpretable assessment of probabilities and a dynamic evaluation of the model and its associated assumptions.

Yet, in the context of man-machine interaction, we need a more general framework than that one afforded by Bayesian analysis alone. We need formal descriptions of the student, M_s, information-gathering experiments, and the methods that tell us how to select them and evaluate their effect. If we wish to display simului dynamically as a function of the user's ongoing performance in order to influence it, we need a topology on inputs, outputs and the relations between them.

Some of these components have been included in mathematical learning models; e.g., Markov chains have been used to describe simple paradigms such as paired-associate learning. Although they allow first-order dependency between observations, in general, they represent stationary processes with fixed, time-invariant, learning rates. They do not include a formal description of the stimuli, the student's responses, and their interrelations.

Optimization, Adaptation, and Individual Differences

Smallwood (1967, 1971) extended the use of Markovian models of learning in an attempt to obtain optimal computer-directed instruction through the application of dynamic programming as formulated by Howard, 1960. Following Howard's technique, whenever the student is in a "state of knowledge" s_i, the teaching system can choose among K_i teaching actions, such as display, test, or stop.

When some action k is selected, the student is assumed to transit from state s_i to state s_j according to the given probabilities (p_{ij}^k) and the system receives a reward (r_{ij}^k). The objective of the teaching system is to maximize its expected reward by selecting an optimal sequence of actions.

This procedure is based on some strong assumptions that are untenable in most instructional contexts. There are three basic problems that this approach ignores. First, the states of knowledge and the transitions between them are not directly observable. Second, the exact form of the model, i.e., the number of states, and the transition probabilities are not known a priori. Furthermore, these probabilities change with different students at different times. Finally, as mentioned previously, the construct "states of knowledge" does not allow concrete interpretation.

Smallwood and Sondik (1973) developed a partial solution to the first problem: the student's states need not be observed directly. Instead, the probability of a given state occupancy is computed from the student's response history. It is more difficult to solve the second problem. We cannot use statistical sampling methods to estimate the probabilities and identify the models (see page 50 for a discussion of identification) because these would require us to calculate response averages over all students and stimuli.

A subsequent report (in preparation) will discuss a Bayesian solution to the problem of estimating parameters for each individual student and different classes of stimuli. This dynamic estimation procedure, investigated by Martin (1967) and Offir (1972), is more realistic than static estimation techniques such as maximum likelihood, minimum chi-square, etc. It is not entirely satisfactory, however, because some assumptions, such as stationarity, have to be maintained.

Briefly, this Bayesian solution assumes that the model's parameters, e.g., the transition probabilities, are random variables from some distribution of performance rates. The distribution represents the dispersion of performance characteristics over all students for a given class of problems. Another way to look at the same idea is to consider the distribution over all students to be composed of an infinite mixture of individual performance distributions. Performance rates (probabilities) are updated individually and are compared with the updated overall students' distribution. Based on results from this comparison, it should be possible to predict performance and generate individual tutorials.

The choice of the initial overall students' distribution, or prior distribution, is motivated primarily by mathematical expendiency, or a lack of better alternative choices. The updating process, however, is intuitively appealing and easy to compuce. As more students are tutored by the system, the distribution stablizes and the system improves its own performance. This evolutionary improvement is called adaptation.

A Toy Example of Generative Remedial Feedback

In this section we use a simple example to illustrate how to generate remedial feedback in a tutorial sequence. This generative procedure is based solely on the subject matter M_m, empirical evidence about the difficulty of the subject matter, and an arbitrary approach to remediation for this particular task, M_t. However, there is no explicit model of the student, M_s, and the tutoring procedure does not improve with time. Hence, according to the present definition the procedure is not adaptive.

The example should be familiar to anyone who has tutored young students to add two single digit numbers, e.g., 3 + 5 = ?. In general, the tutor wants the student's response, y, to be equal to c, the correct response to "a + b = ?" If $y \neq c$, then the tutor generates a remedial problem, a' + b', = ? where either $a' \neq a$ or $b' \neq b$ and a' + b = y. The new problem should prompt the student to discover a contradiction--if his new answer, y', is correct, i.e., if y' = c' = a' + b'. In addition, the tutor chooses a' and b' in such a way that c' is slightly less difficult than c. There is some empirical evidence to suggest that difficulty is directly proportional to the size of the difference between the operands, a and b (Groen and Parkman, 1972).

A simple servomechanistic algorithm that accounts for difficulty is described in the next paragraph. The following is a typical remedial sequence that can be generated by it (T corresponds to tutor and S for student):

T:	What is 6 + 3?	S:	8
T:	No, try 5 + 3 instead	S:	7
T:	No, try 4 + 3	S:	7

(The student should note the contradiction that it is impossible that both 4 + 3 and 5 + 3 are equal to 7).

T:	Yes. If 4 + 3 = 7, then what is 5 + 3?	S:	8
T:	Yes. What is 6 + 3?	S:	9

The algorithm that is used to generate the previous sequence is derived in the following way: Goal: If y is the student's response to the problem a + b = ? and $y \neq a + b$, generate a' + b' such that a' + b' = y and $|a' - b'| \leq |a - b|$. Solution: adding and substracting the equation and inequality above, we obtain respectively, $(a'+b'+|a'-b'|)/2 \leq (y+|a-b|)/2$ and $(a' +b'-|a'-b'|)/2 \geq (y-|a-b|)/2$. In other words, max (a', b') $\leq (y + z)/2$ and min (a',b') $\geq (y + z)/2$; z is the greatest integer less than or equal to |a-b| such that the quantity (y ± z) is even. Then a' = (y + z)/2 and b' = (y - z)/2. When the response is wrong for a+a then the next item is (a-1) + (a-1). Stop when $1 + 1 \neq 2$.

A General Optimal-Control Problem

Consider the system described in Figure 4. Box A represents a controller (a teacher) and Box B, an object (a student) to be controlled. At each time, t, the controller selects an input, x_t, in order to cause B to perform in a way that either satisfies some performance criteria, $D = (d_1, d_2, \ldots d_r)$, or optimizes a function W(D,V), of the desired criteria D and the output sequence, $V = (y_1, y_2, \ldots, y_N)$, where N is the last step that may be unknown and would have to be optimally determined.

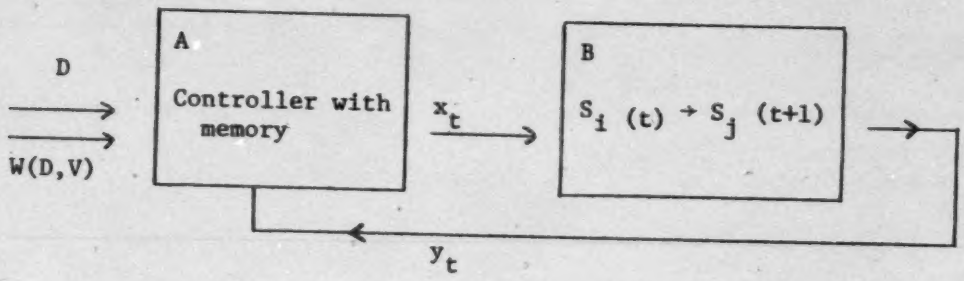


Figure 4. A schematic diagram of a general control system

There may be costs associated with each input, x_t, and costs or rewards associated with the performance indices measured on B. In addition, there may be time and costs constraints that further restrict the choice of inputs. The internal structure of B is described by states, $S = \{s_1, s_2, \ldots, s_c\}$, and the transitions between them, $s_i \rightarrow s_j$. It is usually assumed that B has a finite memory that consists of a finite length of past input-output pairs. The controller may have a similar memory. The output of B, y_t and its associated probability, $p(y_t)$, are determined by B's state and the present input.

The purpose of control is to alter the dynamical behavior of an "object" in accordance with precisely stated objectives. Two problems must be solved to achieve this purpose:

1. We must get a mathematical description of the dynamical properties of the object to be controlled. It includes:

   a. Equations that describe the dynamical behavior of the object as a function of its initial condition and of the control-sequence applied.

   b. Equations that describe all measurable outputs as a function of the control-sequence applied and the object's initial conditions.

2. We must find an algorithm for accomplishing the desired controlled behavior. It is based on:

a. A nonempty set of well-defined set of objectives to be achieved separately, or in combination, and expressed in terms of the object's performance indices.

b. A nonempty well-defined set of feasible controls.

c. A nonempty well-defined set of costs (rewards) and constraints associated with each input and output.

Usually, control and system theorists consider the first problem, model building, to be outside the domain of their investigation. It is assumed that the model is obtained from measurements or known physical laws, and its actual determination is left to the practitioner in the field. It is assumed that one specified version of the model represents the actual process by which the values of the observed quantities have been determined. It often happens, however, that two or more different structural versions of the model lead to the same input-output external behavior, or the same probability distribution thereof. The model is called nonidentifiable if its formal (internal) characteristics cannot be completely determined from the data, however numerous they may be. Much of the work in adaptive control theory is intended to come to grips with this problem of identifiability and related issues (observability, reconstructability, etc.). To date, we can delineate some classes of deterministic and stochastic systems that can be identified and other classes that cannot. The problem is not solved in general.

The theory of optimal control has reached a certain level of maturity. We now possess such theoretical tools as Pontryagin's maximum principle and dynamic programming to derive optimal control systems given the necessary information for the problem (Kalman et al., 1969). In principle, when all the information about the system is available, the optimal control problem can be solved. In reality, "perfect" information is rarely available. This is especially the case in man-computer systems, where the system must collect the missing information during the actual "controlling" of the man.

When the structure and the transition laws are not available and must be inferred from the input/output data, we are faced with an adaptive-control problem. Mathematically, this problem involves the optimization of a random functional with a probability distribution not known in advance.

In general, when we attempt to "control" the behavior of a human user instead of a physical object, we introduce a whole new dimension of uncertainty into the problem. Much of the information needed to affect the behavioral state of each user, especially with untried subject-matter, is usually unavailable. Therefore, the system itself must learn while it

earns, that is, collect information and estimate parameters while trying to control. In order to obtain the information it needs to resolve various uncertainties, the system conducts a series of sequential adaptive experiments; i.e., each experimental sequence, except the first, is determined on the basis of some preceding control-observation pairs. The ways in which the system probes the user's behavioral space should minimize the expenditure of resources, e.g., cost and time, and satisfy the situational constraints, e.g., total computer time available per day, number of terminals, etc.

Summarizing briefly, in addition to objectives, costs and constraints, the makeup of any adaptive system must include the following:

1. Models. The models are formal descriptions of the user, the subject matter and the interaction between the user and the subject-matter.

2. Experiments. The experiments are procedures for drawing conclusions about the latent behavioral characteristics of the user. This is done by selectively applying input (stimuli) sequences and observing the resulting output (response) sequences. The system may have to use different types of experiments in order to construct the latent characteristics of the user.

3. Controls. The controls refer to the system's actions that cause designed behavioral changes in accordance with precisely stated objectives. Usually, the control action selected must also be experimentally informative.

The following section provides a formal discussion of models, controls, and experiments and their interrelationships.

FINITE-STATE SYSTEMS: MODELS, EXPERIMENTS, AND CONTROLS

General System Theory

The notions of a system and a state of a system have been used extensively in many scientific disciplines. Indeed, according to the dictionary definition of a system, almost everything is a system of one kind or another. The purpose of this section is to make the notion of a system and its state a bit more precise. In the next section we posit one class of systems, Stochastic Finite-State Systems, as a candidate for adaptive interaction. Toward the latter part of this section, we discuss different kinds of experiments with finite-state systems and their usage. The results documented in this section have been compiled from several sources; the main ones are the texts by Kalman et al (1969) and Zadeh and Polak (1969).

Inputs, Outputs, Systems and Their States

Box B in Figure 5 represents the object we wish to control. At each moment of time $t \in T$ (reads t in T), the object receives some input (x_t) and emits some output (y_t). We restrict our considerations to events which are taking place in discrete time, i.e., T = {0, 1, 2...}. In general, the output of B depends, deterministically or probabilistically, both on the present input and some past history of input-output information in a way that determines the present output and affects future outputs.

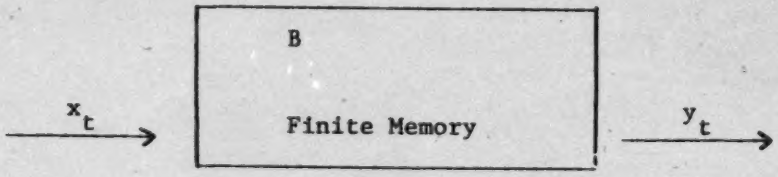


Figure 5. An object with a finite memory

To illustrate these concepts, begin with two sets, X and Y, containing two input and two output elements, respectively. Let X = {a, b} and Y = {0, 1}. We now observe a subset of all input-output (I/O) pairs of length three, (Table 1).

Table 1

Input-Output Pairs of Length 3

1.	(aaa,010)	7.	(abb,011)	13.	(bba,100)
2.	(aaa,101)	8.	(abb,101)	14.	(bba,110)
3.	(aab,010)	9.	(baa,101)	15.	(bba,010)
4.	(aab,101)	10.	(baa,001)	16.	(bbb,101)
5.	(aba,010)	11.	(bab,101)	17.	(bbb,110)
6.	(aba,100)	12.	(bab,001)	18.	(bbb,010)

From the first two strings we see that the object B emits the output (010) or (101) in response to the input (aaa). We demand that any subsequence of an observable I/O sequence must also be observable. For example, for string (2) we require that the subsequences (a,0), (a,1), (aa,01) and (aa,10) must be observable. Let us denote this closure requirement by C. A set of I/O sequences ordered in time and satisfying the C condition is called an abstract object. Since the dependence of output on input is not unique, e.g., we observe the outputs (010) and (101) to the same input (aaa), an abstract object is a relation rather than a function.

In our example, it is possible to show that all the I/O pairs of length, L, greater than three can be completely determined from the pairs of length three. Consider the pair (bab,101) and now apply the fourth input (a). There are two possible outputs (1010) or (1011). The closure property C implies that the pair (baba,1011) is inadmissible, however, since (aba,011) is not listed in Table 1. When the I/O pairs of length L completely determine the I/O pairs of larger lengths, the abstract object is said to have a finite memory of length L. (A formal definition of finite memory of stochastic finite-state systems will be given in the next section.) A system is an abstract object or a collection of abstract objects combined in some way. Figure 6 depicts a system S, which is composed of two abstract objects A_1 and A_2 combined in series.

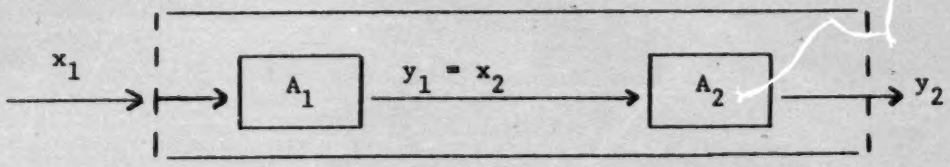


Figure 6. A system of two objects in series

Here, $A_1 = \{(u_1,v_1)\}$, $A_2 = \{(u_2,v_2)\}$,
where u_i is an input sequence of input elements of A_i, $u_i = x_{i_1},x_{i_2},..,x_{i_m}$ and v_i is an output sequence of output elements of A_i, $v_i = y_{i_1}, y_{i_2}\cdots, y_{i_m}$, $i = 1,2$.

We now seek an equivalent state representation, i.e., we want to construct the state space of the system in terms of the I/O pairs listed in Table 1. Informally, this can be done by grouping together I/O pairs into bundles that have common characteristics. Each bundle is then tagged and the tags constitute the states of the system. We use our example (Table 1) to illustrate the rules for bundling.

Let (u^0u^1,v^0v^1) be an arbitrary I/O pair; for each i, u^i and v^i $(i = 0,1)$ are equal-length sequences of inputs and outputs, respectively. (u^1,v^1) is the I/O pair which follows the pair (u^0,v^0). The pairs (u^0,v^0) and (u^1,v^1) are called, respectively, the left and right truncates of the pair (u^0u^1,v^0v^1). Finally, let us denote the tag of the bundle B. We now mention four bundling conditions.

1. If (u^0u^1,u^0v^1) belongs to the bundle B_s, so does the left truncate (u^0,v^0). Example: if (abb,011) belongs to B_s, so do (ab,01) and (a,0).

2. If (u^0u^1,v^0v^1) of length L belong to B_s, then the right truncate (u^1,v^1) (of length $\leq$ L) belong to a common bundle $B_{s'}$. Example: If (u^0u^1,v^0v^1) = (aa,01), (ab,01), (aaa,010), etc. and (u^0,v^0) = (a,0) belongs to B_1 then (a,1), (b,1) and (aa,10) etc. belong to B_2.

3. If (u,v) belong to B_s and (u,v') also belongs to B_s then v' = v. That is, the output must be unique for a given input within each bundle.

4. Every I/O pair must belong to at least one bundle.

The three bundles listed in Table 2 can be derived from the sample of Table 1. It is easy to verify that this bundling satisfies the consistency conditions 1 to 4 listed above.

Table 2

Bundles of I/O Pairs Listed in Table 1

Input	B_1	B_2	B_3
a	0	1	0
b	1	1	0
aa	01	10	01
ab	01	10	01
ba	10	10	00
bb	10	11	01
aaa	010	101	010
aab	010	101	010
aba	010	100	010
abb	011	101	011
baa	101	101	001
bab	101	101	001
bba	100	110	010
bbb	101	110	010

Table 2 can be condensed now to a state-diagram (Figure 7).

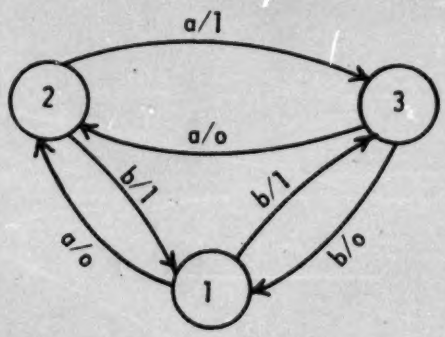


Figure 7. The state diagram corresponding to Table 2

The states are denoted by the nodes (1), (2), and (3). When an input is applied, the system B moves from one state to another and emits an output. For instance, when B is in state (2) and the input "a" is applied, B emits "1" and moves to state (3); when "b" is applied, B emits "1" and moves to state (1).

Let us briefly summarize our example. We started by observing samples of I/O pairs of length three. Assuming that our object had a finite memory of length three we listed the I/O pairs of lengths one and two. Using the four consistency rules, we bundled the I/O pairs into three states. Finally, we described the transition map between the states by means of a state-diagram.

From the uniqueness condition (3), it should be clear that the knowledge of the state at time t and the input sequence u, starting at t, determines the output uniquely at time t+1. From the continuation condition (2), it is clear that the next state is determined by the previous state and the input u. We can combine these observations into the following equations:

$$s_{t+1} = F(s_t, x_t) \qquad (1)$$

$$y_{t+1} = G(s_t, x_t) \qquad (2)$$

$$t = 0,1,2...$$

where s_t denotes the state of S at time t. Equation 1 is called the state-equation and (2) is called the input-output relation. When F and G are single-valued functions, the system is called deterministic; otherwise it is called nondeterministic. (Do not confuse probabilistic systems with nondeterministic ones.) When only finite number of states are sufficient to specify the I/O behavior, the system is called a finite-state system.

The problem of constructing a dynamical system from I/O data is known as the problem of realization and is really the problem of scientific model building. We saw that the formal conditions for associating states with a system are quite straightforward. However, effective solution for the realization problem is more difficult and is available only in special cases. In the general case, F may not be completely determined for the whole time (and consequently neither is G). Scientists usually avoid this problem by assuming that the behavior of the system is constant for all t; that is, if $s_t = s_k$ and $x_t = x_k$, then $s_{t+1} = s_{k+1}$.

Identifiability of Deterministic Systems

An important question which often arises in practice is the question of identification. Assuming that our bundling method satisfies the four consistency conditions specified above, the question is whether the bundles are unique or whether it is possible to have different state representations which exhibit the same I/O behavior. The problem is not solved in general. However, one additional condition would guarantee the identifiability of a deterministic system (see Theorem 1 below). In certain classes of finite-state probabilistic systems (to be defined later), additional conditions are necessary. The problem is not solved in the general case of probabilistic systems.

Example

In Figure 8.a below we changed the state-diagram depicted in Figure 7. Here state (1) is not reachable from either state (2) or (3).

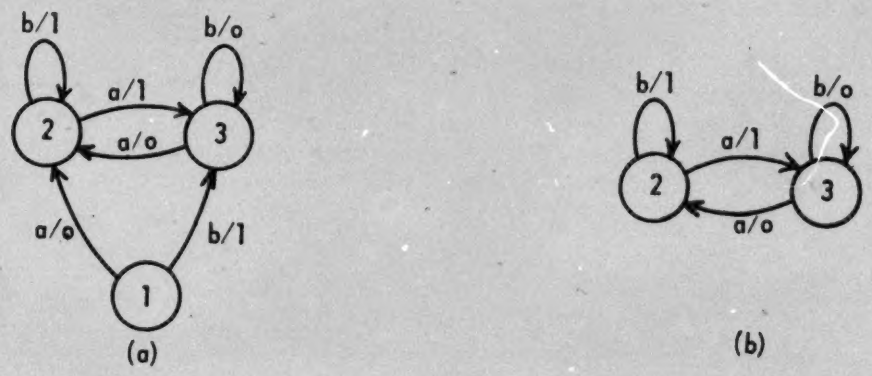


Figure 8. Two weakly equivalent systems.

The reader should verify that equivalent I/O behavior results from the state-diagram of Figure 8.b. That is, the state-diagrams of Figure 8.a and 8.b exhibit the same I/O behavior, yet the system in (b) has only two states.

We say that a state s is reachable if an input string from every other state exists that will transfer the system to state s. The system is said to be strongly connected if every state is reachable. Thus, the system described in Figure 8.b is strongly connected and the one described in Figure 8.a is not.

Theorem 1. A finite-state deterministic system which satisfies the consistency conditions 1 to 4 (see page 27) is identifiable if it is strongly connected.

We are now ready to give some formal definitions.

Deterministic Finite-State Systems: Equivalence and Distinguishability

Definition 1. A finite-state (deterministic) machine M is defined by the following five components:

1. S - a nonempty set of internal states, $S = \{s_1, s_2, \ldots, s_c\}$;
2. X - a nonempty set of input alphabet, $X = \{x_1, x_2, \ldots s_p\}$;
3. Y - a nonempty set of output alphabet, $Y = \{y_1, y_2, \ldots, y_q\}$;
4. F - a transition function which maps all ordered pairs (s_i, x_j) into S, i.e., $F: S \times X \rightarrow S$;
5. G - an output function which maps all ordered pairs (s_i, x_j) into Y, i.e., $G: S \times X \rightarrow Y$;

We write $M = (S,X,Y,F,G)$. A system is defined by a machine M and its starting state s_0, written (M, s_0).

The system operates in discrete time only; that is, the state and output at time t+1 are expressed by equations.

$$s_{t+1} = F(s_t,x_t)$$

$$y_{t+1} = G(s_t,x_t), \quad t = 0,1,2,\ldots$$

Informally, if the next state and output depend only on the previous state and output and not on the time of application the system is called time-invariant or stationary.

The notion of state-equivalence is fundamental to all further developments in this paper. Let U^k and V^k be the sets of input and output sequences, respectively, of length smaller or equal to k; X* and Y* are the set of all finite input and output sequences respectively. Extending the notion of input-output relation we write $s = F(s_i,u)$ and $v = G(s_i,u)$ where u is in X*, v is in Y*, and s_i is the starting state.

Definition 2. Two states s_i and s_j are said to be k-equivalent if for all u in U^k, $G(s_i,u) = G(s_j,u)$; otherwise s_i and s_j are said to be k-distinguishable. If two states are k-equivalent for all k, then they are called equivalent otherwise they are called distinguishable.

Clearly, if two states are k-equivalent they are j-equivalent for all $j \leq k$, and if they are k-distinguishable, then they are j-distinguishable for all $j \geq k$.

Example

Consider the following state-diagram, Figure 9.

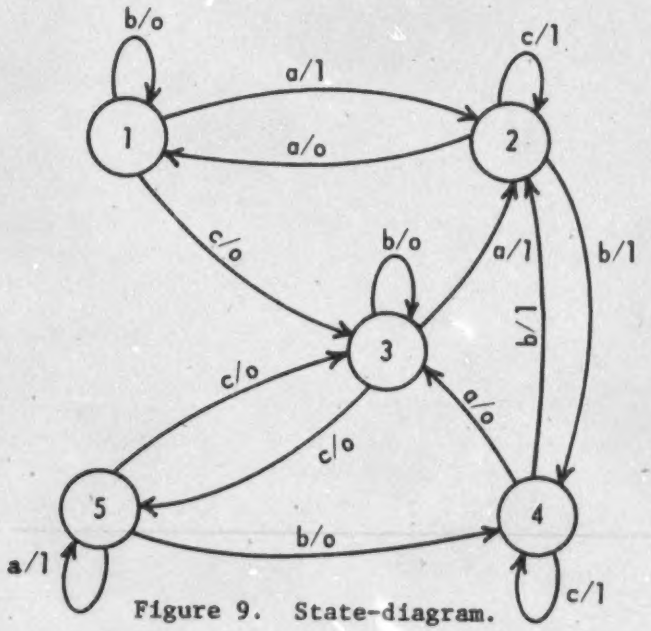


Figure 9. State-diagram.

We can partition the state space s into equivalence classes, E_k, such that two states are in the same equivalence class, if and only if they are k-equivalent. Then for our state-diagram (Figure 9) we have:

E_1 = {1,3,5} , {2,4}

E_2 = {1,3} , {2,4} {5}

E_3 = {1} , {3} , {5} , {2,4}

E_4 = {1} , {2} , {3} , {4} , {5}

In other words, states 1, 3, 5 are 1-equivalent and states 2 and 4 are 1-equivalent. States 1 and 3 are 2-equivalent, but 5 and 1 (or 3) are 2-distinguishable; for example, starting at state 5 and fed with the input (ba) the system emits (00) whereas starting at states 1 or 3 the system emits (01) to the same input. All states are 4-distinguishable.

Definition 3. A system is said to be in reduced form if any two states in its state space are distinguishable, i.e., there are no equivalent states.

Definition 4. Two systems M and N are state equivalent if and only if to each state in M there corresponds (at least) one equivalent state in N and vice versa. Otherwise M and N are distinguishable.

Let $E_1, E_2, \ldots, E_r$ be the k-equivalence classes of a system M which is not in reduced form. In order to construct a new system M' which is in reduced form, we let the state space S' or M' consist of one and only one state from each E_i of S. Then M and M' are state equivalent and M' is the reduced form of M. Among the systems which are state-equivalent to a given system M, those having the smallest number of states are called the reduced-forms of M. As we shall see, the most important questions in system theory can be answered, if answerable at all, for reduced forms only.

Example

The system M' depicted in Figure 10.b is a reduced form of the system M described in Figure 10.a. M and M' are state equivalent. It should be clear that there is no way of discovering by external observations whether the given system is M or its reduced form M', except under special conditions which are described in the next section.

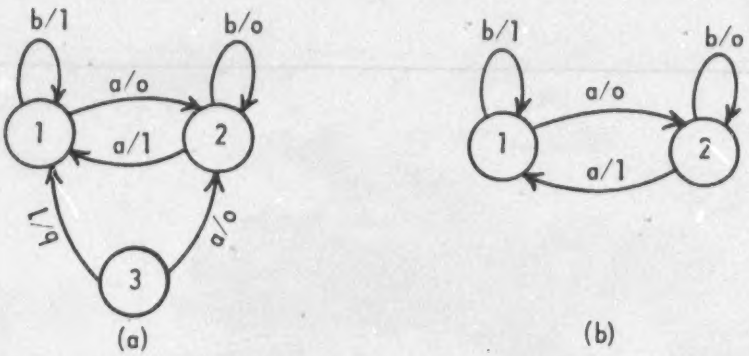


Figure 10. A system M and its reduced form M'.

Experiments with Finite-State Deterministic Systems

An experiment is a procedure of drawing conclusions about the internal structure of the system by applying input sequences and observing the resulting output sequences. The experimenter may have to use different kinds of experiments in order to construct the internal structure of the system. Some types of experiments are described below.

1. A simple experiment. Given that the system is in an unknown state, the experimenter applies an input x_t and observes the output y_t.

2. A preset experiment. The experimenter applies an entire input string, u, determined in advance, and observes various outputs v.

3. An adaptive or branching experiment. The experimenter applies q input subsequences, u_1, u_2,...,u_q, where each subsequence, except the first, is determined on the basis of the preceding observations, i.e., u_{i+1} is a function of (u_i, v_i).

4. A multiple experiment of size n. The experimenter applies n inputs u_1, u_2, ..., u_n to n systems which are assumed to be identical and start at the same state.

Various types of experiments can be combined to form new types; e.g., a multiple adaptive experiment is an experiment of types 3 and 4 combined.

In all types of experiments, the length of the experiment plays a crucial role. The length of the experiment is defined to be the total number of input symbols required for the execution of the experiment.

Most experiments documented in behavioral literature are preset, multiple (and not adaptive) experiments of some arbitrary length. This is unfortunate since quite often it turns out that, even in the case of deterministic systems, some experiments are insufficient or inappropriate to construct the internal structure of the system. We will give examples, but first let us describe some important objectives for experimentation namely: controllability, observability, reconstructibility, and identifiability.

1. A state s of a finite-state system is controllable if the experimenter may bring the system from s to any desired state, s_d. That is, s is controllable if, for any s_d, there exists an input string u such that $s_d = F(s,u)$. A finite-state system is controllable if and only if all its states are controllable.

2. A finite-state system M is initial-state determinable if, no matter what the initial state s_0 is, it is possible to perform some experiment so that s_0 can be determined from the records of the I/O sequences.

From our previous examples it is clear that the system can be initial-state determinable only if it is in reduced form. It is easy to show that a finite-state system is in reduced form if and only if it is initial-state determinable by (at most) an infinite multiple experiment. Consequently, we have the following definition:

3. A finite-state system is observable if and only if it is in reduced form. A diagnosing experiment determines the initial state of the system.

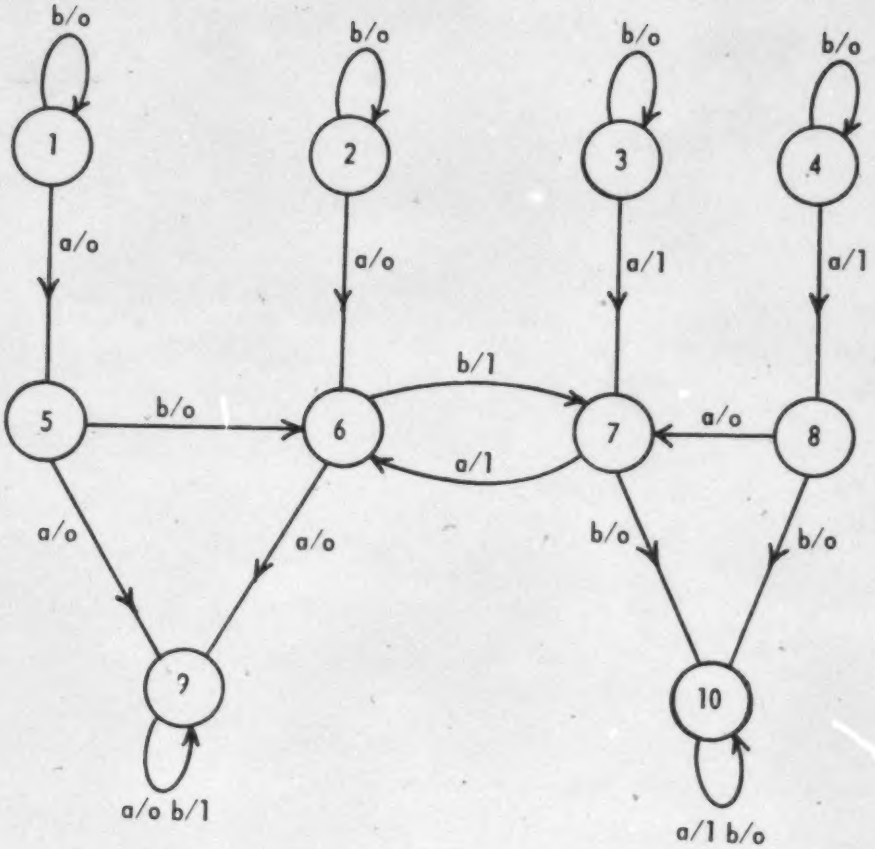
4. A finite-state system M is final-state determinable if, for a given set of initial states, it is possible to perform an experiment such that the state attained at the end of the experiment can be deduced from the records of the I/O sequences. A final-state determinable system in reduced form is said to be reconstructible. A homing experiment determines the final state of M.

Suppose that M belongs to a finite class of distinguishable, reduced-form systems, $C = \{M_1, M_2, \ldots, M_r\}$, all having identical input sets. Then M is said to be identifiable if and only if no state in M is equivalent to any state in M_i, $i = 1,2,\ldots,r$. An identification experiment determines an identifiable system M. A set C is said to be exclusive if no state in any M_i is equivalent to any state in M_j $(j \neq i)$.

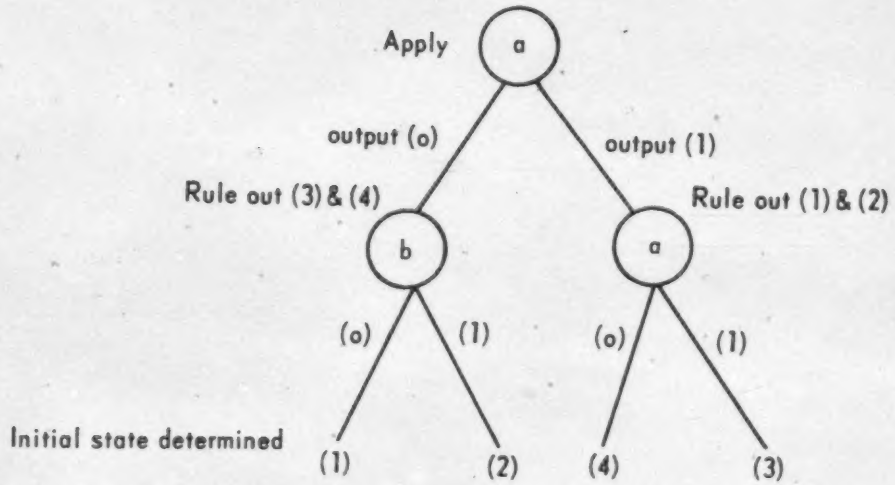
In the remainder of this manuscript, the following abbreviations will be observed: FSS will stand for a finite-state system, RF for reduced form, and IS for the set of states which are known to include the (unknown) initial state.

Example 1: Adaptive vs. preset diagnosing experiment.

Consider the FSS described in Figure 11.a (Gill, 1969). M is RF and IS = {1,2,3,4}. Any minimal diagnosing sequence must initiate with an (a) (rather than with a (b)). The input sequence (aa) confuses between states (1) and (2) and (ab) confuses between states (3) and (4). Therefore, there is no preset diagnosing experiment in this case. A successful adaptive experiment, where each output symbol is observed immediately after each input symbol, is described by the decision tree, Figure 11.b. There are cases where the diagnosing problem cannot be solved by any simple experiment-preset or adaptive. For example, this is the case if IS = {5,6,7,8} in this example, or is the case in example 2.



(a)



(b)

Figure 11. Adaptive diagnostic experiment.

Example 2: Multiple vs. simple diagnostic experiment.

Consider the FSS M described in Figure 12. M is RF. Suppose that any one of the three states (1), (2), and (3) can be the initial state of the system, i.e., IS = {1,2,3}. Our problem is to find a diagnosing experiment; that is, we need to determine the initial state. A simple experiment of any length is insufficient because any input sequence starting with an (a) will result in the same output if the initial state is either (1) or (2); similarly, any input sequence starting with a (b) will confuse states (2) and (3). An adaptive experiment is quite as useless. However, a multiple experiment of size two and length one is a diagnostic experiment; we must have, however, two identical systems to experiment with.

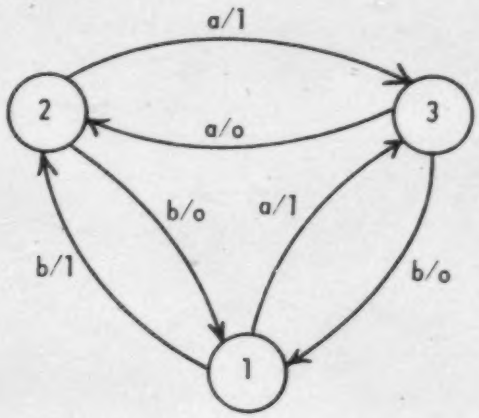


Figure 12. Multiple diagnosing experiment vs. a simple experiment.

In general, if the system has n states with m initial states (i.e., $|IS| = m$) and p input symbols, it is possible to determine whether a simple preset experiment exists and can be found by examining at most p^N input sequences, where $N = \sum_{i=1}^{m} \binom{n}{i}$. In example 2, n = 3, p = 2 and N = 7; therefore, we would have to study at most $2^7 = 128$ sequences of length 7 before we may realize that no simple (preset or adaptive) diagnostic experiment exists. The following theorems are instructive:

Theorem 2. For any RF, n-state FSS there exists a multiple diagnosing experiment of length $\leq n - 1$ and of size $\leq n - 1$.

Theorem 3. The diagnosing experiment for an RF, n-state FSS with $|IS| = m$ is always solvable by a multiple adaptive experiment of length $\leq 1/2(2n - m)(m - 1)$ and size $\leq m - 1$.

The implications of the previous examples and theorems are quite important. Even in the context of deterministic FSS (which is only a special case of the models we have in mine), it is necessary for the system to be in RF in order to be diagnosable. Preset simple experiments are often not diagnosing. Adaptive experiments are better for diagnosing but sometimes not sufficient. Multiple experiments are sufficient for diagnosing but require identical systems with identical initial states. From theorems 2 and 3 above, we immediately see that, if the size of the initial set $|IS| > 2$, an adaptive multiple experiment is longer in length. The results for homing experiments are more encouraging as we see from the next two theorems.

Theorem 4. An RF, n-state FSS with $|IS| = m$ is always final-state determinable by a simple preset experiment of length $\leq (n - 1)(m - 1)$. (If every pair of states is k-distinguishable, this length is $\leq k(m - 1)$.)

Theorem 5. An RF, n-state FSS with $|IS| = m$ is always final-state determinable by a simple adaptive experiment of length $\leq 1/2(2n - m)(m - 1)$ and iteration index $m - 1$. (The iteration index denotes the number of successive subsequences.)

Intuitive implications of these theorems to adaptive CAI systems that are based on finite-state models would be to try to decrease the number of possible initial states of performance for each user and use adaptive homing experiments to determine future states of performance.

Before concluding this section on experiments, let us briefly return to the problem of identification. Suppose that a black-box contains one of the two systems described by Figure 8. There exists no simple experiment whereby the experimenter can decide which system the box contains. However, it is possible that the two systems are indistinguishable under a simple experiment and yet distinguishable under a multiple experiment, that is, an experiment in which more than one copy of the black box is available. For example, a multiple experiment of size 2 is sufficient for determining the systems described in Figure 8. If two copies of system 8.a are available, applying the input (a) to one box would produce a (0) and the input (b) would produce a (1) (when the system starts in state 1). These responses could not be obtained if the same inputs were applied to two identical copies of system 8.b. We say that the two systems are weakly equivalent if they are indistinguishable under a simple experiment. Two systems are strongly equivalent if they are indistinguishable under a multiple experiment. In general we have the following theorems.

Theorem 6. A finite set of RF, strongly-connected, distinguishable FSS's, all admitting the same input, is exclusive.

Theorem 7. Let $C = \{M_1, M_2, \ldots, M_r\}$ be an exclusive set where each FSS M_i $(i = 1,2,\ldots,r)$ has at most n states. Then, there exists a simple preset identification experiment of length $\leq (2n - 1)(nr - 1)$.

Theorem 8. If a FSS M is strongly connected, its I/O alphabets and the bound to the order of its state set are known, then the reduced form of M can always be determined by a simple preset experiment.

Up to this point we have been concerned primarily with the basic relationships between inputs, outputs, and the states of the system in the context of deterministic finite-state systems. We outlined some types of experiments that might help us get a mathematical description of the system we wish to control. Before we turn our attention to the question of optimal control we must consider the question of representation of input, output, and costs. Mathematically, the question of controllability is dependent directly on the representation of the system; whereas some representations of a system may be optimally controllable, others may not. Therefore, we must not confuse a system with its representation.

The Structure of Input, Output and Cost

System theorists are concerned usually with the derivations obtained for classes of abstract models rather than with the questions of model validity or the meaningful assignment of abstract quantities to real-life objects. By and large, it is the practitioner in the field who bears the responsibility for extracting the salient features of the system, interrelating, ordering, and measuring them. Much of this process is problem dependent and hence arbitrary, subjective, and quite often inconsistent from a strict mathematical view point.

Nonetheless, any decision (input, control) can be selected only from a preconceived set of decision alternatives or a feasible set of measurable quantities. Similarly, an acceptable topology must be determined for the output and the cost spaces. For example, in CAI systems we may employ ordinal scales for presentation (input, stimuli) and reinforcement variables, and a continuous interval scale (e.g., response time) for the output.

Optimal control theories (for a special case see next section) usually require that the state, control, and cost spaces possess "nice" topological properties, and also that the state-transition function and cost functional satisfy certain continuity and differentiability conditions (e.g., Falb and Polak, 1969). When these requirements are satisfied, it is possible to obtain necessary (Pontryagin's maximum principle) and sufficient (Hamilton-Jacoby theory) conditions for optimality. Unfortunately, unlike typical control models deterministic and stochastic FSSs have discrete components and thus lack the desired properties.

It is possible, however, to establish some rapprochment between finite-state system and control theories. Arbib (1966, 1967) suggests one way for introducing continuity into discrete sets. This can be done by defining a tolerance relation which is reflexive and symmetric. A set becomes a tolerance space if its elements satisfy the tolerance relation. It is possible then to impose structures on the sets of input and output strings, X* and Y*, so that they become tolerance spaces together. Intuitively, this means that small structural differences between inputs (i.e., input strings satisfying some tolerance relation) induce only small differences in the output strings. This in turn may provide the required continuity and the possibility of using some modified forms of optimal control theories.

Although we are not familiar with examples where these concepts have been actually applied, one could forsee CAI applications where they might be used and afford a means for error analysis as well as optimum control.

Optimal Control Problem of Finite-State Systems

We are now in a position to define an optimal control problem for deterministic FSSs; a similar problem will be formulated later for stochastic FSSs. Our purpose is to merely illustrate some key concepts.

Informally, the idea is very simple. We are given a FSS with a known state-transition function and a known input-output relation. We are also given a cost space with some acceptable topology on it (e.g., the set of integers). At any instant of time k, our immediate cost is a function of the control, x_k, and the state of the system. Our overall cost is a recursive function (e.g., the sum) of our immediate costs. In addition, there are initial and terminal constraints stated as functions with the state as an argument; the simplest example of such constraints is that the system must start in a fixed initial-state and must terminate in a fixed target-state. With these assumptions, our problem is to determine a control sequence (of minimal length) such that the total cost is minimized and the initial and terminal constraints are met.

More precisely, let $M = (S,X,Y,F,G)$ be a FSS as before. Recall that X* and Y* are the sets of all finite input and output strings. The mappings F and G are extended in the following way:

$F: S \times X^* \to S$ and $G: S \times X^* \to Y^*$ that is, there exists an $s' \in S$ and a $v \in Y^*$ such that $s' = F(s,u)$ and $v = G(s,u)$ for all $u \in X^*$ and $s \in S$. Let e be the empty control such that $F(s,e) = s$. Let S_f denote the subset of all terminal states which are acceptable. s_0 denotes the initial state. The acceptable subset of input strings can now be denoted as the set $A = \{u \mid F(s_0,u) \subseteq S_f\}$ i.e., the set of all controls that transfer the system from the initial state s_0 to the final acceptable subset of states S_f. In addition, let C be the cost space with elements c; c_0 is the cost of the initial state s_0.

Definition A of a finite-state sequential decision process (sdp) is the system $D = \langle M, S_f, s_0, c_0, \$ \rangle$ where M, S_f, C, c_0 are as defined above and $\$: CxSxX^* \rightarrow C$ is the cost function such that $\$(c,s,e) = c$ and $\$(c,s,u_1 u_2) = \$(\$(c,s,u_1),F(s,u_1),u_2)$.

The cost function $\$$ is assumed to be a recursive function on X^*. In other words, the total cost $J(u) = \$(c_0,s_0,u)$ can be calculated in a finite number of steps. The sets of optimal control policies are now defined: $O(D) = \{u_i \varepsilon A \mid (\text{for all } u_j \varepsilon A)\ (J(u_j) \geq J(u_i)\ j \neq i)\}$. To solve the sdp problem means to obtain the control sequence $u_i \varepsilon O(D)$. A minimal length sequence $u_i \varepsilon O(D)$ is the optimal control policy.

If certain additional conditions are satisfied, the problem falls within the general class of dynamic-programming problems. However, this does not imply the existence of a general algorithm for solving an arbitrary sdp problem (Ibaraki, 1972). The above model is unrealistic for a CAI application not only because the system is deterministic but also because the states are assumed to be observable.

In the next section we introduce the definition of a stochastic FSS and the control problem for it.

STOCHASTIC FINITE-STATE SYSTEMS: MODELS AND OPTIMAL CONTROL

The psychological literature contains many examples which make use of simple stochastic models (e.g., Markov and Linear Models) in the context of learning theories, perception and other theories. Also in existence are several examples which demonstrate the use of such models in the context of instruction. By and large, however, these models are too restricted in their form to be realistically useful for either prediction or control.

In this section we present the rudiments of stochastic finite-state system (SFSS) theory and its application of optimal control. The model we consider is a stochastic generalization of the deterministic model discussed in the previous section as well as a generalization of the familiar Markovian models. More specifically, we would like to illustrate some powerful tools for optimal-instruction which may be used (1) to account for contingencies and structural differences of stimuli and responses, (2) to deduce state equivalence, occupancy and transition, (3) to predict the future-probabilistic behavior of a system, (4) to determine the minimal length of observation sequence needed for various inferential tasks, and (5) to construct a model from probability sequences generated by certain classes of systems.

The technical issues concerning experimentally-based inference (e.g., diagnosis and identification) are quite involved. Therefore, the topic will be mentioned here only briefly but will be discussed along with other important problems (individual differences and adaptiveness) in a separate article. The section is based primarily on the work by Carlyle (1969); the material on finite-memory and inference is based on the doctoral dissertation by Lin (1973).

Definitions and Basic Relations

First, we generalize the concept of a deterministic FSS.

1. Definitions

Definition 1. A quadruple $M = (X, Y, S, \{A(y|x)\})$ is a stochastic sequential machine if and only if:

(i) X, Y and S are nonempty finite sets (the input, output, and internal-states sets, respectively).

(ii) $\{A(y|x)\}$ is a nonempty finite set containing $|X|$ times $|Y|$ square matrices $A(y|x) = [a_{ij}(y|x)]$ of order $|S|$ such that $a_{ij}(y|x) \geq 0$ for all i and j, and $\sum_{y \in Y} \sum_{j=1}^{|S|} a_{ij}(y|x) = 1$.

A Stochastic Finite-state system (M,Π) consists of a machine M and a probability distribution $\Pi = \{\pi_s : s \in S\}$ i.e., $\Sigma_s \pi_s = 1$. Π is called an initial distribution over the states of the system. Specifically, if the system starts in state s_i then $\Pi = s_i$ denotes a degenerate distribution, i.e., $\pi_i = 1\ \pi_j = 0$ for $s_j \neq s_i$. In this case we write (M,s_i).

To explain, the system starts in state s with probability π_s. $a_{ij}(y|x)$ is the probability of the machine going to state j and printing the output $y \in Y$, given it had been in state i and fed with $x \in X$; i.e., $a_{ij}(y|x) = p(y,s_j|x,s_i)$. A Markov matrix, A(x), summarizes the behavior of a SFSS with respect to all of its output, that is, $A(x) = \sum_{y \in Y} A(y|x)$. If the system admits only one input, i.e., $X = \{x\}$, it is called an autonomous system. A SFSS with a single input and output is called a homogeneous Markov chain.

Definition 2. A stochastic input-output relation is any function $p(v|u)$ whose domain is the set $\{(u,v): u \in X^*, v \in Y^*\}$ and whose range is the interval [0,1], such that (1) $p(v|u) \geq 0$, (2) $p(e|e) = 1$, and (3) $p(v|u) = \sum_{y \in Y} p(vy|ux)$; (e is the empty symbol).

If there exists a SFSS (M, Π) whose I/O relation coincides with p, then p is said to be of a finite-state type and (M, Π) is called a finite-state representation or realization for p. Note that the definition requires $\sum_{y \in Y} p(y|x) = 1$ for all $x \in X$.

Next, we calculate the probability of some important events. The probability that the system prints the sequence of length L, $V = y_{j_1} y_{j_2} \ldots y_{j_L}$ and transits to state s, given it had been in state r and fed with the sequence $u = x_{i_1}, x_{i_2} \ldots x_{i_L}$ is defined by the a_{rs} entry in the matrix

$$A(v|u) = [a_{rs}(v|u)] = A(y_{j_1}|x_{i_1})\ A(Y_{j_2}|X_{i_2}) \ldots A(Y_{j_L}|X_{j_L}).$$

If follows that

$$a_{rs}(ay|ux) = \sum_m a_{rm}(v|u)\ a_{ms}(y|x). \tag{4}$$

If the initial distribution is Π, then the probability that the SFSS will print v in response to u is given by $P_\Pi(v|u) = \Pi A(v|u) 1$ where 1 is the column vector with unit entries.

The probability that the system is at state m immediately after it had been fed with the k^{th} input is given recursively by the m^{th} entry of the vector.

$$\underset{\sim}{\Pi}(k+1) = \underset{\sim}{\Pi}(k)\, A(y_{j_k} | x_{i_k}) \;/\; [\underset{\sim}{\Pi}(k) a(y_{j_k} | x_{i_k}) \underset{\sim}{1}], \tag{6}$$

where $\underset{\sim}{\Pi}(k)$ is the distribution vector over the states before the k^{th} input is presented.

The expression (6) is important since it enables us to calculate the probability of unobservable state occupancy in terms of observable I/O behavior. Using (6) we have $P_{\underset{\sim}{\Pi}}(v|u) = \prod_{k=1} P_{\underset{\sim}{\Pi}}(k)\ (y_{i_k} | x_{i_k})$. (7)

2. Equivalence

Definition 3. Two SFSS's M and M' are state-equivalent if, to every state s_i of M, there corresponds a state s'_j of M' and vice versa such that $P^{M}_{s_i}(v|u) = P^{M'}_{s'_j}(v|u)$ for all I/O pairs v|u of length ≥ 1. (s_i is a degenerate initial distribution vector.)

Definition 4. Two SFSS's $(M, \underset{\sim}{\Pi})$ and $(M' \underset{\sim}{\Pi})$ are distribution equivalent if $P_{\underset{\sim}{\Pi}}^{M}(v|u) = P_{\underset{\sim}{\Pi}'}^{M'}(v|u)$ for all I/O pairs v|u of length ≥ 1.

Two SFSS's M and M' are k-equivalent if their I/O relation agree for all I/O pairs of length $\leq k$, and k-distinguishable if their I/O relation disagree for some pairs $\leq k$. Clearly k distinguishability implies (k+1) distinguishability and (k+1) -equivalence implies k equivalence.

3. Reduced Forms

Among the machines which are state-equivalent to a given machine M, those having the smallest number of states are called the reduced forms of M. A machine M is said to be in a reduced form if it has no equivalent states. In particular, there are very simple construction rules which provide a recipe of how to merge or lump together equivalent states, so that if M is an $|S|$-state stochastic machine with at least one pair of equivalent states, then there exists an $(|S|-1)$-state machine which is state-equivalent to M. As in the deterministic case, it is inferentially important to get rid of all the redundant equivalent states, that is, to obtain a reduced form of M. Unlike the deterministic case, a stochastic machine does not possess a unique reduced form but a continuum of distinct reduced form. However, when M belongs to a special class of machines (states calculable, see page 47), then it has a unique reduced form.

4. The Number of Observations Needed to Determine Equivalency and Accessibility

Suppose that we have two identical SFSSs except for their initial distributions, or equivalently, one SFSS and two possibly different distributions: $(M,\underset{\sim}{\Pi})$ and $(M,\underset{\sim}{\Pi}')$. How many observations do we have to take in order to decide whether their distributions are equivalent, i.e., $P_{\underset{\sim}{\Pi}}(v|u) = P_{\underset{\sim}{\Pi}'}(v|u)$? The answer is given in the following theorem:

Theorem 1. If M has $|S|$ states, then $P_{\underset{\sim}{\Pi}}(v|u) = P_{\underset{\sim}{\Pi}'}(v|u)$ for all pairs $v|u$ of length $(|S|-1)$ implies $P_{\underset{\sim}{\Pi}}(v|u) = P_{\underset{\sim}{\Pi}'}(v|u)$ for all pairs $v|u$.

For two different SFSSs the answer is given by:

Theorem 2. Two SFSSs $(M,\underset{\sim}{\Pi})$ and $M', \underset{\sim}{\Pi}')$ are equivalent if and only if they are $(|S|+|S'|-1)$-equivalent.

We see, then, that in order to ascertain whether two systems will have identical probabilistic behavior, it is enough to look at initial probability sequences of lengths smaller than the sum of their states order.

Definition 5. A state s_i of a SFSS $(M,\underset{\sim}{\Pi})$ is accessible if there exists an I/O pair $v|u$, $(e|e)$ included, such that the probability of reaching this state $\pi_i(v|u) \neq 0$. (Accessibility is the stochastic analogue of the deterministic reachability.)

Theorem 3. If state s_i is an accessible state of a SFSS with $|S|$ states, then there exists an I/O pair of length $\leq (|S|-1)$ such that $\Pi_i(v|u) \neq 0$.

Consequently, in order to determine what states are accessible, calculate the state probability $\pi_i(v|u)$ for all the states i of interest and all sequence $v|u$ of length $\leq |S|-1$.

It is often desirable to employ only connected systems, where all the states are accessible. For any SFSS $(M,\underset{\sim}{\Pi})$, it is possible to construct an equivalent connected SFSS by eliminating all nonaccessible states from the system.

Special Classes of Stochastic Systems

1. Probabilistic Automaton

In calculating the output-distribution $P_{\underset{\sim}{\Pi}}(v|u)$ in the previous sections, we assumed that all the responses were available at intermediate

times. Often, this assumption may not be fulfilled because either responses are unobservable or may have not yet occurred. We briefly consider here a particular case of SFSSs where only the final output is of interest.

Definition 6. The probability $p(y|ux) = \Pi A(u)A(y|x)1$ where $A(u) = A(x_{i_1}) A(x_{i_2})...A(x_{i_{L-1}})$ and x is the $L^{\tilde{}th}$ input is called a final-output distribution (f.o.d.)

A SFSS for which the choice of observables is restricted to the final output y is called a probabilistic automaton.

We are interested in utilizing the f.o.d. for the purpose of statistical inference. Namely, we would like to determine the set of all input sequences such that the f.o.d. is greater than any ε such that $0 < \varepsilon < 1$. The set of all input sequences satisfying this property, denoted by $I(\varepsilon,y) = \{u:p(y|u) > \varepsilon\}$, is called an input-event. We say that the event $I(\varepsilon,y)$ is defined by the automaton (M,Π) with a cutpoint ε and an output y.

The immediate question that comes into mind is, what is the relationship between the concepts of input-event and that of system equivalence? The answers are quite interesting.

First, we note that $p(y|ux,v) = p(vy|ux) / p(v|u)$.

Therefore, we can define a new event -- the input-output event -- as $I/O(\varepsilon,y) = \{(ux,v):p(y|ux,v) > \varepsilon\}$. In words, this is the set of all sequences (ux,v) of a general SFSS having an f.o.d. $p > \varepsilon$. In this case though, the entire response sequence is observable. We say that two SFSS are I/O event equivalent if they have the same sets of I/O events as defined above. Similarly, two SFSS are I-event equivalent if they possess the same sets of I events. The following theorem relates the notion of general equivalence to the concept of event equivalence:

Theorem 4. (a) Two SFSS are equivalent if and only if they are I/O-event equivalent.

(b) Deterministic systems are equivalent if and only if they are input-event equivalent.

(c) There are distinguishable SFSS which are I-event equivalent.

Clearly then, I-event equivalence of SFSSs is not, in general, sufficient to assure system equivalence except in the special case of input state-calculable systems (see next section).

2. State Calculable Systems

In the case of deterministic systems, it is possible to determine the final state uniquely from the initial state and the input sequence. In the general case of SFSSs this is impossible because a SFSS can transit with some probability to more than one state with a given I/O pair. We may, however, restrict our attention to a particular class of SFSSs, namely, state-calculable systems, in order to obtain degenerate terminal distributions, unique reduced forms, and various other analytic and inferential benefits.

We recognize three types of state calculable machines, as defined below; b. and c. are special cases of a.

Definition 7.

a. An observable state calculable machine (OSC) is a machine whose transitions are completely determined by the I/O pair and the initial state. That is, $p(y,s'|s,x) = 0$ if $s' \neq f(s,x,y)$. [Every row is the matrix $A(y|x)$ contains at most one nonzero entry.]

b. An input state calculable machine is a machine whose transitions are completely determined by the input and the initial state. That is, $p(y,s'|s,x) = 0$ if $s' \neq f(s,x)$.

c. An output state calculable machine is a machine whose transitions are completely determined by the output and the initial state, that is, $p(y,s'|s,x) = 0$ if $s' \neq f(s,y)$. Note that deterministic machines are trivially a special case of state-calculable machines.

The following theorem describes the unique inferential properties possessed by state-calculable systems which do not generalize outside this class:

Theorem 5.

a. Let (M,s) be a state-calculable SFSS with $|S|$ states. Then every terminal distribution vector $\underset{\sim}{s}A(v|u)/P_{\underset{\sim}{s}}(v|u)$ is degenerate. That is, if a state-calculable system starts at a given state with probability 1, then it is possible to determine uniquely the final state.

b. Every state-calculable machine has a unique state-calculable reduced form.

c. Let (M,Π) and (M',Π') be two SFSSs which are input-state calculable. If $(M,\underset{\sim}{\Pi})$ and $(M'\underset{\sim}{\Pi}')$ are input-event equivalent, then they are equivalent and conversely. (If M and M' are OSC machines, this theorem need not hold.)

3. Output Independent Machines (OIM)

In CAI and in other psychological applications, it is most likely that the transition and the output are independent events (c.f., Atkinson et al., Chapter 3, 1965, Offir, 1973); the output may follow or precede the state transition. This is explicated in the following definition:

Definition 8. An output-independent machine (OIM) must satisfy equation 8.a or equation 8.b:

$$A(y|x) = A(x)\ D(y|x) \qquad (8.a)$$

$$A(y|x) = D(y|x)\ A(x), \qquad (8.b)$$

where $A(x)$ is stochastic, and $D(y|x)$ is a diagonal matrix such that $\sum_{y\in Y} D(y|x) = I$, the identity matrix. The process generated by an OIM is called partially observable unless one of the following equations is satisfied in addition.

$$d_{jj}(y|x) = p(y|s_j,x) = p(y|x) \qquad (9)$$

The process generated by the OIM of type (9) is called nonobservable since the output is independent of the state of the system and its observation will not yield new information about the state.

$$d_{jj}(y|x) = p(y|s_j,x) = 1 \text{ if } y = f(s_j) \text{ and } 0 \text{ otherwise} \qquad (10)$$

The process generated by (10) is called completely observable since any output y_j corresponds uniquely to a state s_j; the states transitions are thus known by observing the output.

$$d_{jj}(y|x) = d_{jj}(y) \qquad (11)$$

The process generated by (11) is said to have a probabilistic output (note that the output is independent of the input).

From the above definitions, it can be immediately seen that if a machine M is OSC and, in addition, satisfies (8.a), then it is either completely observable or input-state-calculable, if it satisfies (8.b), then it must be input-state-calculable. Now, if two systems (M,Π) and (M',Π) are completely observable, it is trivial to decide whether they are equivalent. Secondly, by theorem 5 (part 3), they are equivalent if they are input-event equivalent when they are input-state-calculable and from the proceeding argument when they are both OIM and OSC. Particularly, this means that we can check whether OSC systems which satisfy equation 8 are equivalent by comparing their input events, $I(\varepsilon,y)$.

4. Completion and System Representation of an I/O Relation

In a previous section we defined an I/O relation, $p(v|u)$, (c.f., definition 2), and I/O relations of a finite-state type. Now, consider an I/O relation with arguments $v|u$ of length $\leq n$. A relation of this type is called an initial segment of length n whereas any relation which has a specified initial segment is called a completion of that segment. In general, any initial segment may have many completions.

We are interested in the solution to two problems.

a. Finding an algorithm for determining uniquely the completion of any relation, p, of a finite-state type.

b. Finding, in addition, a finite state-representation to p.

The solution to the first problem is described in theorem 6. The second problem has a solution only in special cases, in particular for the class of state-calculable systems.

Theorem 6. Let p be any I/O relation of a finite-state type which does not require more than $|S|$ states. Then the completion of p can be determined (constructively) uniquely if its initial segment of length $(2|S|-1)$ is given.

The following definition is needed to answer the second question:

Definition 9. The left-derivative of a relation p with respect to the initial segment $[u',v']$ is the function

$$p_{[u',v']}(v|u) = \begin{cases} p(v'v|u'u)/p(v'|u') \text{ if } p(v'|u') > 0 \\ \text{the zero function otherwise} \end{cases}$$

seeking a finite state representation for p we have:

Lemma. Let p be an I/O relation. If the set of all nonzero derivatives of p is finite and contains n different relations, then there exists an n-state machine M - such that $p = p^{M}_{s_1}$.

Now, the question is, are there machines with finite number of distinct derivatives for $p^{M}_{s_1}$? The answer is yes for state calculable machines. This is so because every terminal distribution $s_1A(v|u)/s_1A(v|u)1$ is a stochastic degenerate vector (theorem 5.1). Since there are only $|S|$ such vectors, the relation p_{s_1} has only $|S|$ nonzero derivatives.

We can, therefore, choose a state-calculable machine, M, so that each derivative is identified with a state of M which is reachable from s_1. We thus have the following theorem:

Theorem 7. If an I/O relation p has $|S|$ distinct derivatives, then there exists a state-calculable system (M,s_1) such that $p = p_{s_1}$. (The knowledge of the initial segment of length $2|S|-1$ is sufficient to determine, constructively, a unique state-calculable system representation for p with minimum number of states.)

Identifiability of SFSS With Finite Memory

In previous sections we pointed out that many inferential problems can be solved only when the underlying SFSS is state-calculable. In particular, the problem of deriving a system representation from an I/O relation is unsolvable in the general case but is solvable in the case of state-calculable systems. Usually, however, it is not important to obtain an exact representation but only a representation up to some level of equivalence which would suffice to produce or explain the I/O relation of interest. The obvious question is then whether we can find a general class of systems which are equivalent in some sense to an OSC system and then take advantage of the latter's properties.

It turns out that the answer is affirmative; the class of systems we are seeking is the class of strongly-connected, finite-memory systems with a known input and a known bound on the order of the system's memory. Starting with this class it is then possible to find an "equivalent" OSC system by performing some experiment. It should be noted that this is not a very restricted class of systems, since it is possible to force every machine to behave like a finite-memory machine. In what follows, we list some theorems which lead to the aforementioned conclusion. Similar statements apply to the solution of the diagnosis problem of SFSSs.

Definition 10. A stochastic machine is said to be a finite-memory machine of order m if and only if m is the smallest integer such that the probability $p(y_t|x_t) = f(x_{t-m}, x_{t-m+1},\ldots,x_t, y_{t-m},y_{t-m+1},\ldots y_{t-1})$ for some function f.

Theorem 8. There is a finite procedure to decide whether a given machine is finite-memory of a given order.

One relationship between finite-memory and OSC systems is expressed in the following theorem.

Theorem 9. Let M be a finite memory machine of order m. Then for every distribution Π on M there exists an OSC machine M' such that for some state $s \in M'$ (M,Π) is distribution equivalent to (M',s).

Theorem 10. If an I/O relation $p(v|u)$ can be represented by a finite-memory SFSS, then it can be represented by an OSC SFSS.

The concept of a terminal distribution graph plays a fundamental role in the solution of the identification problem.

Definition 11. Let M be a finite-memory machine of order m. Let Π be an initial distribution such that $\Pi_i = \Pi A(v|u)_i / \Pi A(v|u)1$ exists, where $(v|u)_i$ is the i^{th} I/O sequence of length $m' \geq m$. Then, the terminal distribution graph (TDG) of order m' of M, denoted G(M,m'), is a directed graph with n nodes each corresponding uniquely to Π_i $i = 1,2,\ldots n$. The node N_i is directed to the node N_j with label $\emptyset_{ij}(y|x)$ if and only if $p_{ij}(y|x) = A(y|x)1 \neq 0$ for any Π_i and Π_i is defined by $y'v|x'u$ while Π_j is defined by $vy|ux$ for some I/O pair $y'|x'$. The set of terminal distributions is denoted by $\Gamma(M,m') = \{\Pi_1, \Pi_2, \ldots \Pi_n\}$.

Definition 12. Let G(M,m) and G(M',m') be two TDGs. The node N in G and the node N' in G' are said to be equivalent if and only if, for every I/O sequence $(y_t y_{t-1} \ldots y_1 | x_t x_{t-1} \ldots x_1)$, the path in G beginning at N with the sequence of labels $\emptyset(y_t|x_t)\emptyset(y_{t-1}|x_{t-1})\ldots\emptyset(y_1|x_1)$ have the same probability; i.e., if and only if $p(y_t y_{t-1} y_1 | x_t x_{t-1} \ldots x_1) = p'(y_t \ldots y_1 | x_t \ldots x_1)$. The probability is zero if the path does not exist. The graphs G and G' are said to be equivalent, $G(m,m) \equiv G(M',m')$, if and only if for every node in G there is an equivalent node in G' and vice versa.

Clearly the equivalence of TDGs is similar to that of SFSSs when the latter are represented by state graphs. The following theorems are important in order to obtain an identification experiment:

Theorem 11. Let M be a finite-memory of order m. Then $G(M,m_1)$ is equivalent to $G(M,m_2)$ for any m_1, $m_2 \geq m_1$ if M is strongly connected.

Theorem 12. Let M be a finite-memory machine of order m. Then its TDG of order m', for an $m' \geq m$, represents M up to a distribution-equivalence if every state, regarded as a distribution, is contained in some class of $\Gamma(M,m')$ and up to state-equivalence if every class of Γ contains some state.

Theorem 13. For every finite-memory machine M, the TDG for M specifies M up to a certain degree of equivalence (depending the conditions satisfied as in Theorem 12) to an OSC machine.

An identification experiment exists which enables us to construct the TDG (nodes and transition probabilities within a desired accuracy level) of an OSC system which is equivalent, to a certain degree, to the strongly-connected, finite-memory system we are experimenting with.

Optimal Control of Stochastic Finite-State Systems

The optimal control problem of completely-observable stochastic machines was formulated and solved for special cases of Markov chains by Howard (1960). The reader should consult Howard's text for an illuminating introduction to this subject. In the present chapter we formulate the optimal control problem of partially-observable SFSSs, where the state occupancy is not directly observable. We possess an algorithm that directs the selection of optimal controls (inputs) at each step of the remaining control stages--assuming a finite number of these stages. The algorithm itself is not described here but will be detailed in a subsequent, more technical, paper. It is based on the fact that the optimal payoff function is a piecewise-linear and convex function of the state probabilities $\Pi(k)$.

Let (M,Π) be a SFSS; in particular, M can be an output-independent machine (OIM). Recall that the state probabilities at the instant k+1 are given recursively by

$$\Pi(k+1) = \Pi(k)A(y_{j_k}|x_{i_k}) / P_{\Pi(k)}(y_{j_k}|x_{i_k}) \quad , \quad k = 1,2,... \qquad (12)$$

Let $M = (X,Y,S\{A(y|x)\})$ be a stochastic sequential machine.

Let $R(y|x) = [r_{ij}(y|x)]$ be a reward matrix;

$r_{ij}(y|x)$ is the reward accrued when the system emits y in response to the control x and transits from state i to state j. It is assumed that only controls and responses can be observed and nothing else. The immediate reward vector $q(x) = [q_i(x)]$ is defined by

$q_i = \sum_y \sum_j a_{ij}(y|x) r_{ij}(y|x)$. When n steps remain in the process, the optimal control equation is written:

$$V(\Pi_n) = \max_{x_n} \Pi_n [q(x_n) + \sum_{y_n} A(y_n|x_n) V(\Pi_{n-1}|y_n,x_n)] \quad , \text{ and}$$

$V(\Pi_0) = \Pi_0 q_0$, where q_0 is a given terminal reward vector over the states of the system.

It turns out, [4], that Eq. 15 can be written as

$V(\Pi_n) = \max_m \Pi_n \lambda_n^m$ and is a piecewise linear and convex function; $m = f(\Pi,x,y)$ is finitely valued and the vectors λ_n^m cut the simplex $\{\Pi : \sum \pi_i = 1\}$ into equivalence regions $Ex_n = \{\Pi : V(\Pi_n|x_n) = \Pi_n \lambda_n^{m(x_n)}\}$. The optimal control at stage n , x_n^* is determined by maximizing.

$$V(\Pi_n) = \max_{x_n} [q(x_n) + \sum_{y_n} A(y_n|x_n)\lambda_{n-1}^m] \quad , \text{ subject to}$$

$$\Pi_n A(y_n|x_n)[\lambda_{n-1}^{m^*} - \lambda_{n-1}^m] \geq 0 \quad , \text{ and}$$

$$\Pi_n \lambda_n^{m^*} \geq \Pi_n \lambda_n^m \quad , \text{ where } m^* = f(\Pi,x^*,y) \quad .$$

SUMMARY AND CONCLUSIONS

The probabilistic finite-state approach, described in this report, represents a new systematic way for constructing training and aiding systems. Its usefulness is derived from its ability to formally capture the structural relationships among instructional units relative to each individual problem solver. However, in order to develop systems that are fully adaptive, i.e., systems that -- in addition -- improve their own performance with usage, further work must be undertaken to provide them with probability assessment and updating techniques. Such techniques already exist.

REFERENCES

Arbib, M. A. Automata theory and control theory: A rapproachment. Automatica, 1966, 3, 161-189.

Arbib, M. A. Tolerance automata. Kybernetik, 1967, 3, 22-233.

Arbib, M. A. The Metaphorical Brain. New York: Wiley Interscience, 1972.

Atkinson, R. C., Bower, G. H., & Crothers, E. J. An introduction to mathematical learning theory. New York: Wiley & Sons, 1965.

Berztiss, A. T. Data Structures. New York: Academic Press, 1971.

Carbonell, J. R. AI in CAI: An artificial intelligence approach to computer-assisted instruction. IEEE Transactions on Man-Machine Systems, MMS-11, 1970, 190-201.

Carlyle, J. W. Stochastic finite-state system theory. In L. A. Zadeh and E. Polak, System Theory, New York: McGraw Hill, 1969.

Chomsky, N. A review of B. F. Skinner's Verbal Behavior Language, 1959, 35, 26-58.

Chomsky, N. Aspects of the theory of syntax. Cambridge, Mass.: The MIT Press, 1965.

Falb, P. L. & Polak, E. Conditions for optimality. In L. A. Zadeh and E. Polak System Theory. New York: McGraw Hill, 1969.

Fine, T. L. Theories of probability. New York: Academic Press, 1973.

Fu, K. S., & Booth, T. L. Grammatical inference: Introduction and survey - Part I. IEEE Transactions on Systems, Man, and Cybernetics, SMC-5(1), 1975, 95-111.

Gill, A. Finite-state systems. In L. A. Zadeh and E. Polak, System Theory, New York: McGraw Hill, 1969.

Goldberg, A. Computer-assisted instruction: The application of theorem proving to adaptive response analysis. Technical Report 203, Institute for Mathematical Studies in the Social Sciences, Stanford University, 1973.

Groen, G. J., & Parkman, J. M. A chronometric analysis of simple addition. Psychological Reveiw, 1972, 79, 329-343.

Hewitt, C. Description and theoretical analysis (using schemata) of Planner, Ph.D. Thesis, MIT, 1972.

Howard, R. A. Dynamic programming and Markov processes. Cambridge, Mass.: The MIT Press, 1960.

Ibaraki, T. Solvable classes of dynamic programming. Systems, computers, controls, 1972, 3(b), 23-30.

Kalman, R. E., Falb, P. L., & Arbib, M. A. Topics in mathematical system theory. New York: McGraw Hill, 1969.

Knuth, D. E. The art of computer programming, Fundamental Algorithms, 1968, Reading, Mass.: Addison-Wesley.

Lighthill, J. General survey of artificial intelligence. London: British Science Research Council, 1972.

Lin, H. A. Identification of finite-memory stochastic sequential machines. Unpublished Ph.D. dissertation, UCLA School of Engineering, 1973.

Manna, Z., & Waldinger, R. J. Toward automatic program synthesis. Communication of the ACM, 1971, 14(3), 151-165.

Martin, J. J. Bayesian decision problems and Markov chains. New York: Wiley & Sons, 1967.

Miller, G. A. Needed: A better theory of cognitive organization. IEEE Transactions on Systems, Man and Cybernetics, SMC-4(3). 1974, 95-97.

Miller, G. A., & Chomsky, N. Finitary models of language users. In R. D. Luce, R. R. Bush, & E. Galanter (Eds.) Handbook of Mathematical Psychology, New York: Wiley, 1963.

Narendra, K. S., & Thathacher, M. A. L. Learning automata - a survey. IEEE Transactions on Systems, Man and Cybernetics, SMC-4(4). 1974, 323-334.

Newell, A., Shaw, J. C., & Simon, M. A. Report on a general problem solving program. Proceedings of International Conference Information Processing, Paris: UNESCO House, 1959, 256-264.

Newell, A., & Simon, H. A., Human problem solving. Englewood Cliffs: NJ, Prentice Hall, 1972.

Nilssen, N. J. Problem-solving methods in artificial intelligence. New York: McGraw Hill, 1971.

Offir, J. D. Stochastic learning models with distributions of parameters, Journal of Mathematical Psychology, 1972, 9(4), 404-417.

Offir, J. D. Automaton model of performance. Journal of Mathematical Psychology. 1973, 10(4), 353-363.

Paz, A. Introduction to probabilistic automata. New York: Academic Press, 1971.

Pitts, W. H., & McCulloch, W. S. How we know universals: The perception of auditory and visual forms. Bulletin of Mathematical Biophysics, 1947, 9, 127-147.

Quillian, M. R. The teachable language comprehender. Communication of the ACM, 1969, 12, 459-476.

Simon, H. A., & Lea, G. Problem solving and rule induction: A unified view. In Gregg, L. W. (Ed.) Knowledge and Cognition. Potomac, MD: Lawrence Erlbaum Associates, 1974.

Smallwood, R. D. A decision structure for teaching machines. Cambridge, Mass.: MIT Press, 1962.

Smallwood, R. D. Quantitative methods in computer directed teaching systems. Final Report Nonr-225(84), Stanford, CA: Institute in Engineering-Economic Systems, Stanford University, 1967.

Smallwood, R. D. The analysis of economic teaching strategies for a simple learning model. Journal of Mathematical Psychology, 1971, 8, 285-301.

Smallwood, R. D., & Sondik, E. J. The optimal control of partially observable Markov processes over a finite orizon. Operations Research, 21(5), 1973, 1071-1088.

Suppes, P. Stimulus response theory of finite automata. Journal of Mathematical Psychology, 1969, 8, 327-355.

Suppes, P., & Morningstar, M. Computer-assisted instruction at Stanford, 1966-1968. New York: Academic Press, 1972.

Tversky, A., & Kahneman, D. Availability: A heuristic for judging frequency and probability. Cognitive Psychology, 1972, 5(2), 151-175.

Weiner, N. Cybernetics. Cambridge, Mass.: The Technology Press, 1948.

Wexler, J. D. Information networks in generative computer-assisted instruction. IEEE Transactions on Man-Machine Systems, 1970, NMS-11, 181-189.

Winograd, T. Understanding natural language. New York: Academic Press, 1972.

Zadeh, L. A., & Polak, E. System Theory, New York: McGraw Hill, 1969.